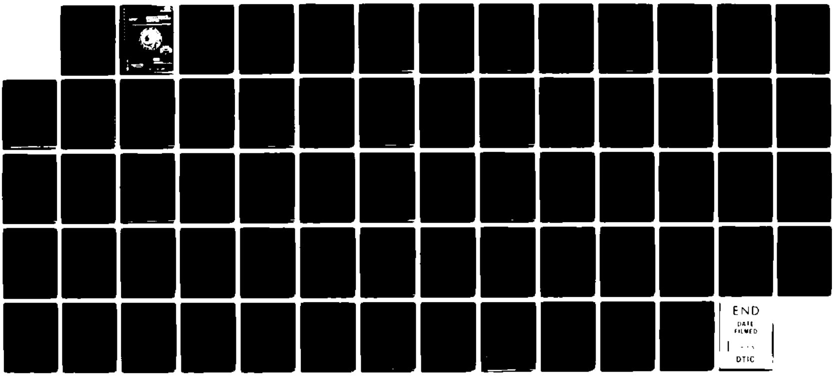
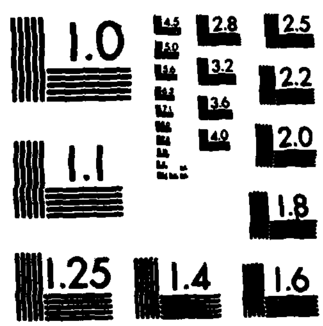
AD-A121 498 DOCUMENTATION OF SOFTWARE FOR THE NEIL BROWN INSTRUMENT 1/1
SYSTEMS/NORDA VEL..(U) NAVAL OCEAN RESEARCH AND
DEVELOPMENT ACTIVITY NSTL STATION MS..
UNCLASSIFIED K D SAUNDERS ET AL. JUL 82 NORDA-TN-155 F/G 8/10 NL




1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0

ABSTRACT

Software developed for processing Neil Brown Instrument System/NORDA profiler data is documented in this report. The software includes programs for translating the profiler data from original NBIS format to engineering units in UNIVAC/NAVOCEANO FEB files and for editing and correcting the data subsequently. This report provides complete descriptions of the programs as well as operating information.

ACKNOWLEDGMENTS

This work was sponsored by the NORDA Code 500 Ocean Measurements Program, Dr. Rudy Hollman, Program Manager.

CONTENTS

DOCUMENTATION OF SOFTWARE FOR THE
NEIL BROWN INSTRUMENT SYSTEMS/NORDA VELOCITY/CTD PROFILER

I. INTRODUCTION

This report constitutes both documentation and user manual for software associated with the Neil Brown Instrument Systems/NORDA velocity/CTD profilers. As the software is in a constant state of development, this report is presented in a loose-leaf format, allowing for replacement of outdated software, correction of existing software and addition of new software, and operation instructions.

The initial software was written in HPL for the Hewlett-Packard 9825A calculator by Kim Saunders and Henry Perkins. The basic data collection and display programs were subsequently modified and expanded by James Vega of Computer Sciences Corporation. The 9825A is too small and slow for the data processing envisioned, requiring the use of a large mainframe or super-mini computer.

Vega wrote a set of three programs for the translation of profiler data to engineering units in UNIVAC 1108/NAVO/NORDA Fast Easy Binary (FEB) file format, the conversion of these data to orthogonal, geomagnetic coordinates, and the graphic display of these data. The first two of these programs were subsequently corrected and modified by Saunders.

Fred Hamrick of Computer Sciences Corporation wrote two programs to compute the vertical instrument velocity and correct the vertical velocity. The algorithms for these programs were developed by Saunders and Perkins, first testing in rough form on the HP 9825A.

After the GYRE cruise to the equatorial Atlantic in November-December 1981, it was found that a serious over-ranging problem occurred during occasionally strong down-swings of the instrument. The algorithm and program to approximately correct these errors were developed and implemented by Saunders.

These programs constitute, at present (April 1982), the existing software developed specifically for the profiler. NORDA Code 331 has, however, a growing suite of utility programs for the processing of FEB files. The documentation for these programs is not included in this report, but will constitute a separate report.

A DEC VAX 11/750 super-mini computer is in procurement for the sea-going data processing of the profiler data. New programs to appear in this report will soon be available and will include:

- real-time profiler data aquisition,
- acoustic navigation collection,
- real-time filtering and correction,
- real-time display.

II. SUMMARY OF DATA COLLECTION AND PROCESSING

The Neil Brown Instrument System/Naval Ocean Research and Development Activity 3-Axis Velocity/CTD profiler system was designed for the study of upper ocean mixing and variability. It is capable of measuring three components of velocity, acceleration, and magnetic field, as well as conductivity, temperature and pressure. This system and the initial phases of testing are documented in Perkins et al. (1980) and Saunders et al. (1981).

The profiler system consists of an underwater unit and a deck unit. The data are sensed by the underwater unit, digitized, and then transmitted to the surface via an audio FSK code over a single conductor (sea water return) cable. The signal is led through a winch with slip-rings to the deck unit, where the data are reformatted. There are three signal outputs from the deck unit:

- an audio output for backup on audio tape,
- an IEEE 488 parallel bus and,
- an output for a digital tape recorder (either a Kennedy or Digi-Data).

When the initial test of the profiler was conducted, the physical oceanography branch did not own a 9-track tape drive that could be directly connected to the third output. The branch owned an interface/buffer/recorder system that was compatible with the IEEE 488 bus. The initial data collection program was therefore designed to transfer data directly from the deck unit to the interface/buffer, occasionally breaking into the bus to obtain "snapshots" of the data being recorded. The present version of the data collection program "DATALOGGER" incorporates this design. However, it was recently discovered that because the interface/buffer is only a singly buffered system, about 15% of the data are lost when the buffer is emptied during recording. The solution to this problem is to record the data directly to magnetic tape via the third output.

The data collection program is also designed to produce quasi-realtime plots of the data on one or two printer-plotters and to display numeric data on a Hazeltine 1420 CRT terminal at 9600 Baud.

Once the data have been recorded on digital magnetic tape, they are processed on the UNIVAC 1108 at NAVOCEANO (this will be changed in the near future to a DEC VAX 11/750). The program which translates the data from the NBIS format to engineering units is "TRANSCRIBE." This program will handle data written directly from the deck unit to magnetic tape or tape generated by the IEEE 488-interface / buffer system.

The data should then be plotted and if there appears to be any evidence of over-ranging, the "PREFIX" program should be run. This program makes a close approximation to the velocities when over-ranging is encountered.

Once this is done, either "VFIX1" or "VFIX1-S" may be applied to the data. These programs attempt to correct the vertical velocity by computing and subtracting the instrument velocity from the observed vertical velocity. "VFIX1" accomplishes this by integrating the vertical acceleration, while "VFIX1-S" differentiates the pressure to obtain the instrument's vertical motion.

Two utility programs are also included. "UNORTHOG" is used mainly for testing and debugging purposes for looking at the velocity data in the original acoustic (non-orthogonal) components. "TSERPLOT2" is a general plot package to plot any of the variables in the FEB files against either cycle number or time.

III. DATA STRUCTURES

Two primary structures are involved in profiler data processing: NBIS raw data format and profiler FEB files. The NBIS format is used only during the data collection phase. This is a highly packed format prior to conversion to engineering units. The FEB file structure is the standard file format used in the Physical Oceanography Branches at NORDA and NAVOCEANO (Hallock, 1981). The details of these formats are

given below for reference. The NBIS format is given in Figure 1 (from the NBIS profiler manual, with permission).

The FEB file variables, after the transcription phase are, in order of position in the data array:

No.	Name	Variable (units)
1	PRESS	pressure (decibars)
2	STEMP	slow response temperature (deg C)
3	COND	conductivity (mmho)
4	FTEMP	fast response temperature (deg C)
5	VLOCI1	velocity component 1 (cm/sec)
6	VLOCI2	velocity component 2 (cm/sec)
7	VLOCI3	velocity component 3 (cm/sec)
8	MAGI1	magnetic component 1
9	MAGI2	magnetic component 2
10	MAGI3	magnetic component 3
11	ACCI1	acceleration comp. 1
12	ACCI2	acceleration comp. 2
13	ACCI3	acceleration comp. 3
14	TIME	reference time (dec. days)
15	RELSEC	relative time (sec)

Further processing programs occasionally will change, correct, or replace these variables. For instance, after using PREFIX, variable 14 is replaced with an estimate of the instrument's vertical velocity, determined from the pressure time derivative. It is also possible to extend the number of variables in the FEB file structure, and at present the programs VFIX1 and VFIX1-S extend the number of variables to 17.

The header blocks for the PROFILER FEB files are defined as follows:

ADOC (1) - (31): available for alphanumeric information.
FDOC (1) - sample interval time
(2) -
(3) - start latitude (dec. deg.)
(4) - start longitude (dec. deg.)
(5) - time of start fix (dec. days)
(6) - end latitude
(7) - end longitude
(8) - time of end fix
(9) - maximum pressure of sensor
(10) - cast start time (dec. days)
(11) -
(12) -
(13) - cast end time
(14) -
(15) -
(16) - magnetic variation
(17) - magnetic dip
(18) - ship's speed (kt)
(19) - ship's heading
(20) -
(21) - dry bulb temp (deg. C)
(22) - wet bulb temp (deg. C)

(23) - surface temp (deg. C)
(24) - barometric pressure (mb)
(25) - wind speed (nm)
(26) - wind direction (compass)
(27) - significant wave height (ft)
(28) -
(29) -
(30) -
(31) - start time of profile (day)
(32) - " " " " (hour)
(33) - " " " " (min)
(34) - " " " " (sec)
(35) -
(36) - end " " " (day)
(37) - " " " " (hour)
(38) - " " " " (min)
(39) - " " " " (sec)
(40) -

IDOC (1) - end of profile flag
(2) - cruise number
(3) - station number (id format)
(4) - relative segment number
(5) - series sequence number
(6) - absolute no. of 1st seg. in series
(7) -
(8) -
(9) -
(10) - input tape no.
(11) - year of cast
(12) -
(13) - no. of bad cycles
(14) - no. of profiles in cast
(15) -
(16) -
(17) -
(18) -
(19) -
(20) -

Figure 1. NBIS Raw Data Format

BYTE	PARAMETER	DISPLAY UNITS	LS BIT WEIGHT	FORMAT*
1	Frame Sync.	240 or 015	---	--
2	Pressure LSB	see calibration	see calibration	AC
3	Pressure MSB			
4	Temperature LSB	degree celcius	0.5 m deg. C	AC
5	Temperature MSB			
6	Conductivity LSB	mmho	0.001 mmho	AC
7	Conductivity MSB			
8	Fast Temp. LSB	degree celcius	0.5 m deg. C	AC
9	Fast Temp. MSB			
10	AC Signs	part of temp. and pressure	---	SIGNS
11	Velocity X LSB	16383=1 m/sec	1/16383 m/sec	DC
12	Velocity X MSB			
13	Velocity Y LSB	16383=1 m/sec	1/16383 m/sec	DC
14	Velocity Y MSB			
15	Velocity Z LSB	16383=1 m/sec	1/16383 m/sec	DC
16	Velocity Z MSB			
17	Compass X LSB	ratio only	---	DC
18	Compass X MSB			
19	Compass Y LSB	ratio only	---	DC
20	Compass Y MSB			
21	Compass Z LSB	ratio only	---	DC
22	Compass Z MSB			
23	Acceleration X LSB	g's x 1000	0.001 g	DC
24	Acceleration x MSB			
25	Acceleration Y LSB	g's x 1000	0.001 g	DC
26	Acceleration Y MSB			
27	Acceleration Z LSB	g's x 1000	0.001 g	DC
28	Acceleration Z MSB			
29	Spare-0 LSB**	--	--	DC
30	Spare-0 MSB**			
31	Spare-1 LSB**	--	--	DC
32	Spare-1 MSB**			
33	TOD-0 msec	--	m sec x 10, msec x 1	BCD
34	TOD-1 sec/msec	--	sec x 1, msec x 100	BCD
35	TOD-2 min/sec	--	min x 1, sec x 10	BCD
36	TOD-3 hr/min	--	hr x 1, min x 10	BCD
37	TOD-4 day/hr	--	day x 1, hr x 10	BCD
38	TOD-5 day	--	day x 100, day x 10	BCD

*NOTE 1. AC Format
LSB = 128, 64, 32, 16, 8, 4, 2, 1
MSB = 32768, 16384, 8192, 4096, 2048, 1024, 512, 256

2. SIGNS Format
1, 1, 1, 1, 1, Fast Temp., Temp., Pressure
where 1 = negative, 0 = positive

3. DC Format
LSB = 32, 16, 8, 4, 2, 1, 0, SIGN
MSB = 8192, 4096, 2048, 1024, 512, 256, 128, 64

4. BCD Format
8, 4, 2, 1, 8, 4, 2, 1

** All bits in spare Bytes are set to "1"S.

Figure 1. NBIS Raw Data Format

IV. REFERENCES

Hallock, Z. R. (1980). The Fast and Easy Binary (FEB) File. NAVOCEANO TN 7210-12-80.

Perkins, H. T., K. D. Saunders, G. Appell, and T. Mero (1980). Design and Initial Testing of a Three Axis Acoustic Current Meter. OCEANS 80 Conference Record (IEEE Pub. No. 80CH1572-7), 319-322.

Saunders, K. D., H. T. Perkins, L. Banchero, S. Sova, and J. J Vega (1981). Sea Trials of a Lowered Three Axis Current Meter. OCEANS 81 Conference Record (IEEE Pub. No. 81CH1685-7), 245-249.

APPENDIX A: AN ESTIMATOR FOR VERTICAL INSTRUMENT VELOCITY

Let w_0 and a_0 represent observed vertical components of velocity and acceleration and let p_0 be the observed pressure. Suppose the data to be given over the time interval t_1, t_2

Define the instantaneous vertical velocity of the instrument as

$$w_i(t) = \int_{t_1}^{t} [\alpha\, a_0(t) - \tilde{g} - \gamma]dt + \beta \qquad \text{(A-1)}$$

NOTE: In the program, g was set equal to zero and a correction, ACORR=9.99, was applied to all measured accelerations.

where g is the local gravitational acceleration, α and γ are corrections to the observed acceleration and β is the instrument velocity at $t=t_1$. The values of α, β, and γ are to be determined in an optimal way, as described below. Nominally, $\alpha = 1$ and $\gamma = 0$.

Further define the mean vertical velocity of the instrument during t_1, t_2 on the basis of the corresponding pressure change:

$$\bar{w}_i = \frac{1}{\bar{\rho} g}[p_0(t_1) - p_0(t_2)]/(t_2 - t_1) \qquad \text{(A-2)}$$

Where $\bar{\rho}$ is the mean density of the water.

The quantities α, β, γ are determined by the conditions

$$\int_{t_-}^{t_2} (w_i - w_0)^2\, dt = \text{minimum} \qquad \text{(A-3)}$$

and

$$\int_{t_1}^{t_2} (w_i - \bar{w})\, dt = 0 \qquad \text{(A-4)}$$

That is, w_i is required to resemble w_0 as much as possible and also correspond to the known mean instrument speed.

Formally, the two equations above constitute a linear least squares problem with a side condition. This may be reformulated by the method of Lagrange:

$$\int_{t_1}^{t_2} [(w_i - w_0)^2 + \lambda(w_i - \bar{w})]\, dt = \text{minimum} \qquad \text{(A-5)}$$

Where λ is a *Lagrange* multiplier, the value of which is also to be determined. Differentiating this expression with respect to α,β,γ and λ in turn and equating each of the results to zero, as is required to minimize the expression, results in a linear system of 4 equations for $\alpha,\beta,\gamma,\lambda$. These have the form

$$AX = B \tag{A-6}$$

Where $$A = (A_{ij}) = \left(\int_{t_1}^{t_2} a_{ij}(t)\, dt \right) \tag{A-7}$$

$$X = \begin{pmatrix} \alpha \\ \beta \\ \gamma \\ \lambda \end{pmatrix} \tag{A-8}$$

$$B = (B_j) = \left(\int_{t_1}^{t_2} b_j(t)\, dt \right) \tag{A-9}$$

NOTE: In the program, the last quantity in array B, $\int \bar{w}$, was set equal to the average velocity determined from the total pressure change multiplied by the total time.

The quantities a_{ij}, b_j can be shown to be

$$(a_{ij}) = \begin{pmatrix} f_1^2 & f_1 & -f_1 f_2 & \tfrac{1}{2} f_1 \\ f_1 & 1 & -f_2 & \tfrac{1}{2} \\ f_1 f_2 & f_2 & -f_2 & \tfrac{1}{2} f_2 \\ f_1 & 1 & -f_2 & 0 \end{pmatrix} \tag{A-10}$$

$$(b_j) = \begin{pmatrix} f_1(w_0 + gf_2) \\ w_0 + gf_2 \\ f_2\,(w_0 + gf_2) \\ \bar{w} + gf_2 \end{pmatrix} \tag{A-11}$$

where $$f_1 = f_1(t) = \int_{t_1}^{t_2} a_0(t)\, dt \tag{A-12}$$

$$f_2 = f_2(t) = \int_{t_1}^{t_2} dt = t - t_1 \tag{A-13}$$

Note that even though (a_{ij}) is singular, since the first three rows are multiples of each other, (A_{ij}) is not. Hence, it can be inverted, equations A-6 solved for α,β,γ and the result used in A-1 to find an optimal estimate of the instrument velocity.

APPENDIX B: PROFILER SOFTWARE

PROGRAM: DATALOGGER

PURPOSE: To log data and maintain a real time display of certain variables when the profiler is collecting data at sea. Data are logged by transferring the data stream directly to an IDEAS IEEE 488/CIPHER DATA Interface/buffer. The variables are displayed on a Hazeltine 1420 CRT terminal and plotted on two HP 7242 printer/plotters.

MACHINE: HEWLETT-PACKARD 9825A

LANGUAGE: HPL

AUTHOR: Kim David Saunders, Henry T. Perkins and James J. Vega

FILE LOCATIONS: Tape V002, File 4

INPUT: The program solicits the following:
Cast Number,
Latitude in degrees decimal,
Longitude in degrees decimal.
These are used only for display on the Hazeltine terminal. During normal operation of the program, different display options may be invoked by pressing the f0 function key. The program will then solicit the type of display option desired and the unit to which that type of display is to be directed. The display options are listed after the program listing.

OUTPUT: The output consists of alphanumeric listings of data on the Hazeltine screen and plots of data on either or both printer plotters.

ADDITIONAL INFORMATION:

PROGRAM: TRANSCRIBE

PURPOSE: To convert profiler data from raw form on the original data tapes (or condensed copies) into engineering units with the velocity expressed in orthogonal instrument coordinates. The converted data are stored in FEB file format for ease in further processing.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHORS: James J. Vega, corrected and modified by Kim David Saunders

FILE LOCATIONS: Absolute, Relocatable, and Symbolic Elements - VEGA*LIB.TRANSCRIBE

INPUT: The input parameters are solicited by the program for use in an interactive environment.
The input raw data must be attached to logical unit 10, for example by a series of statements such as:

```
asg,tsj 10,u9s,<<tapeno.>>
move 10,<<nfiles-1>>
```

The solicited input parameters are summarized below for use in a batch environment. All input is in free format.
Lines 1-3: Alphanumeric documentation 42 char/line.
Line 4: Cruise no., station no., absolute no. of 1st segment of the output file, input tape no., year of cast (all integer).
Line 5: Time interval between samples in seconds, zero, starting latitude in decimal degrees, starting longitude, time of fix a start of station in decimal days, ending latitude, ending longitude, ending time, maximum pressure of profile in decibars. (all floating point)
Line 6: Magnetic variation, magnetic dip, ship speed, ship heading (floating point).
Line 7: Dry bulb temperature, wet bulb temperature, surface temperature, wind speed in knots, wind direction in degrees, significant wave height in feet (floating point).
Line 8: Cast start day, hour, minute, second zero, cast end day, hour, minute, second (integer).
Line 9: Sequential file no. (integer)
Line 10: Station Identification No. consisting of a 3 digit station and a 3 digit sequential cast No.
Line 11: Message level 0-9

OUTPUT: File 20 contains the output FEB file.

ADDITIONAL INFORMATION:

<u>PROGRAM:</u> CONVERSION

PURPOSE: To convert from instrument orthogonal to geomagnetic orthogonal coordinates.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: James J. Vega, corrected and modified by Kim David Saunders

FILE LOCATIONS: Absolute, Relocatable, and Symbolic Elements - VEGA*LIB.CONVERSION

INPUT: File 10 - input FEB File.

OUTPUT: File 20 - output FEB File.

ADDITIONAL INFORMATION:

PROGRAM: PREFIX

PURPOSE: To read a FEB file containing raw profiler data, removing points where the vertical velocity has exceeded the limits for the instrument.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: Kim David Saunders (April 1982)

FILE LOCATIONS: Absolute, Relocatable, and Symbolic Elements - VEGA*LIB.PREFIX

INPUT:
Line 1) NUIN1,NSEG1,NSSEG1
Line 2) NOUT
NUIN1 = unit number of imput FEB file
NSEG1 = number of segments to be read
NSSEG1 = number of first segment
NUOUT = unit number of output FEB file

OUTPUT: The output FEB file has the same structure as the input file, with the exception that variable 14 now contains a rough approximation to the vertical instrument velocity estimated from the time derivative of the pressure. The initial time of the cast (in Julian days has been replaced).

ADDITIONAL INFORMATION: The NBIS 3VCTD profiler measures the current relative to the instrument by means of a three axis acoustic velocimeter. The operation of this type of current meter is described in the NBIS Acoustic Current Meter manual. The point of interest, here, is that the apparent velocity along any axis is proportional to the phase difference of the two acoustic pulses, which are, in turn, proportional to the true component of the water velocity along the axis. Thus, in principle, the measured velocity should be mapped onto the interval from about -100 to 100 cm/sec (nominal). In practice, this does not occur exactly, as when the phase of the acoustic signal is near -180 or 180 degrees, the gate opening/closing signals become ambiguous. This results in random output velocities when the true velocity component along the axis is within a small "dead band" of the velocity extremes. This program is designed to search the data for probable occurrences of this over-ranging and to correct (as much as possible) by substituting the projection of the vertical instrument velocity, determined by the time derivative of the pressure, for the components along the q1 and q3 axis.

PROGRAM: VF1X1

PURPOSE: Corrects the vertical component of current velocity as measured by the NORDA 3-component profiler.

INPUT: (Free format)

Line 1 - Input file specifications
IUNR - Unit no. for input
NSEG - No. of segments to process
NSSEG- No. of starting segment
MSGR - Message level for input
Line 2 - Output file specifications
IUNW - Unit no. for output
MSGW - Message level for output

NOTES:

1) The input file is presumed to be in geomagnetic coordinates as produced by program CONVERSION.

2) Input variables are identified by name as follows:

Vertical current speed, uncorrected	- VLOCG3
Vertical component of acceleration	- ACCLG3
Pressure	- PRESS
Time	- RELSEC

3) The output file has the same structure as the input file except that the corrected vertical velocity, named W, is inserted in each data cycle immediately before the uncorrected vertical velocity, and the instrument velocity, named WI, is inserted in each data cycle immediately after the last variable (RELSEC).

METHOD: An estimate of vertical instrument velocity w_i is found from the observed acceleration and pressure. Derivation of w_i is given in the Appendix. The corrected velocity w is then obtained from the observed velocity w_0 by

$$w = w_0 - w_i$$

PRINTED OUTPUT: For each output segment, the following quantities are listed:

- o Start and end times (RELSEC)
- o Start and end pressure
- o $\alpha,\beta,\gamma,\lambda$ (see Appendix for definitions)

MACHINE: UNIVAC 1108

FILE LOCATIONS: Absolute element CODE331*FCHFILE1.VFIX1
Mapping element CODE331*FCHFILE1.MVFIX1

PROGRAM: TSERPLOT2

PURPOSE: To plot time series of profiler variables either versus cycle number or relative time.

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: James J. Vega

FILE LOCATIONS: Absolute, Relocatable, and Symbolic

INPUT: There are two input lines for each subplot:

```
Line 1:  IU, IABSIS, IP, YMAX, YSTP, YMIN, IDEC
Line 2:  if IABSIS=0:  NUMSEG, IBEGIN, CYCIN
         if IABSIS=1:  TSTART, TEND, IPTIME
```

where

```
IU     = logical unit for input FEB file
IABSIS= 0 for cycle number plot
      = 1 for time plot
IP     = position of variable in FEB array
YMAX   = expected max. of variable
YSTP   = labling interval
YMIN   = expected min. of variable
IDEC   = decimation ratio

NUMSEG= number of segments to plot
IBEGIN= first segment to plot
CYCIN = number of cycles per inch

TSTART= start time in seconds (relative)
TEND   = stop time in seconds
IPTIME= location in FEB array of time
```

OUTPUT: File 25 - output intermediate plot file.

ADDITIONAL INFORMATION: 1. The following data should succeed the last data line to ensure proper termination of the program:

99,0,0,0.,0.,0.,0 .

2. This program is designed to use 34 inch ZETA plotting paper. Because of this, the maximum number of variables per plot is 5.

<u>PROGRAM:</u> VF1X1-S

PURPOSE: To read a FEB file containing profiler data and create a new FEB file which contains a corrected vertical velocity computed from the pressure derivative.

MACHINE: UNIVAC 1108

AUTHOR: Fred Hamrick (April 82)

FILE LOCATIONS: Absolute element CODE331*FCHFILE1.VF1X1-S
Mapping element CODE331*FCHFILE1.MVF1X1-S

INPUT: Line 1) IUNR,IUNW,NSEG,NSSEG,NVAR,NUMV

IUNR = unit number for input FEB file
IUNW = unit number for output FEB file
NSEG = number of segments to read
NSSEG= start segment
NVAR = variable number for pressure
NUMV = variable number for vertical velocity

OUTPUT: The output FEB file has the same structure as the input file with two additional variables. The variable WI (instrument velocity) is written as the last variable, and the corrected velocity W=V-WI (where V=measured vertical velocity) is written as the variable immediately before the measured vertical velocity.

ADDITIONAL INFORMATION:

1) Before executing this program, the input FEB file should be interpolated with respect to time. (This time difference is set in the program as variable DELT.) The interpolation may be performed by executing HTP*PROG.ZINTERP.
2) The instrument velocity is computed as

   WI = RHOG1*DPDT
   Where RHOG1 =   = .9955
   and DPDT =   = pressure derivative
   is computed as:

$$DPDT(J) = \frac{3}{K(K+1)\ (2K+1)*DELT} \sum_{i=-K}^{i=+K} (i * P_{j+1})$$

   Where P_j = pressure values
   and K is set to 8 in the program.
   (the first and last K values of DPDT are set to 0).

<u>PROGRAM:</u> UNORTHOG

PURPOSE: To convert from instrument orthogonal coordinates to instrument acoustic axes coordinates. (Velocity only).

MACHINE: UNIVAC 1108

LANGUAGE: FORTRAN V

AUTHOR: Kim David Saunders

FILE LOCATIONS: Absolute, Reloctable and Symbolic Elements- VEGA*LIB.UNORTHOG

INPUT: From terminal (unit 5)

Line 1: NUIN, NSEG, NSSEG
Line 2: NUOUT

NUIN = Input unit No.
NSEG = No. of Segments desired
NSSEG= No. of first segment.
NUOUT= Output unit No.

APPENDIX C: COMPUTER LISTINGS

```
0: "FILE 4 - TAPE ID V002":
1: "Profile data logger-Part 2":ldk 2
2: "LATEST MOD:  02 DEC 1981:1645Z[KDS]":
3: "Added second depth profile subroutine":
4: " Added pressure Corection":
5: "Changed profiler from unit 8 to unit 5 - 8 now res. for disk":
6:
7: "O array controls output options":
8: " [1] contains Hazeltine list option":
9: " [2] contains 7245A time series plot option":
10: " [3] contains 7245A depth plot option":
11: " [4] lists summaries of instrument motion statistics":
12: " [5] produces a second depth profile on unit 703":
13: " [6] produces a plot of one variable vs the other",
14: "Options 3 and 4 disable each other to prevent device conflict":
15:
16: "Function keys":
17: " f0- solicit display options":
18: " F0 start/stop data logging":
19: " F1 Generate end of data file":
20:
21: "Flags":
22: " 0 - if set, log to mag tape; if clear, don't":
23: " 1 - if set, close the data file; if clear, continue logging":
24: " 5 - if the matrices in the DISPLAY routines have been      ":
25: "     - dimensioned, flg 5 is set.                           ":
26: " 6 - if set, the first pass through the TIMESERIESPLOT routine":
27: " 7 - first pass flag for INSTVEL                            ":
28: " 8 - if set, the 2 nd pass through the TIMESERIESPLOT routine":
29:
30: dim A[66,3],O[9],Z[38],T[27],R[0:10],S[1:]
31: dim V$[66,20],V[10,0:10],Q$[3],U[2],T$[60]
32: dim B[3,3],F[4,3],D[6],E[6]
33: "LOAD VELOCITY ORTHOGONALIZATION MATRIX":
34: -.055→F[1,1];-.803→F[2,1];1.58→F[3,1];-.794→F[4,1]
35: -.197→F[1,2];-.634→F[2,2];-.011→F[3,2];.759→F[4,2]
36: .612→F[1,3];.849→F[2,3];-.071→F[3,3];.809→F[4,3]
37: trk 0;ldf 5,V[*];ldf 6,V$;trk 0
38: 0→A;ent "Enter the Cast No.",A[1,1]
39: ent "Enter the Latitude in decimal degrees",A[6,1]
40: ent "Enter Longitude-decimal degrees",A[7,1]
41: cos(52)→A[11,1];-sin(52)→A[12,1]
42:
43: -3→O[1];0→O[3];0→O[2]
44: on err "ERRORHANDLER"
45: dev "nb",520
46: buf "b2",38,1
47: buf "b1",38,1;time 15000;fxd 0
48: "START":if flg1;ldp 7
49: buf "b1"
50: cmd 5,"____"
51: tfr "nb","b1",38
52: jmp rds("b1")=38
53: wait 200
54: buf "b2"
55: cmd 5,"___"
56: tfr "nb","b2",38
57: jmp rds("b2")=38
58: if flg0;gto "next"
59: cmd 5,"_,_"
60: cmd 5,"?T+"
61: "next";for I=1 to 38;rdb("b1")→Z[I];next I
62: if Z[1]=79;cll 8;prt "ERR";gto "START"
63:
64: cll 'BREAKOUT'
```

```
65: for I=1 to 66;A[I,1]→A[I,2];next I
66: for I=1 to 38;rdb("b2")→Z[I];next I
67: if Z[1]=79;cli 8;prt "ERR";gto "START"
68: cll 'BREAKOUT'
69: cll 'OPTIONS'
70: cll 'DERIVE'
71: cll 'DISPLAY'
72:
73: if abs(A[15,1]-A[15,3])<1;gto "START"
74: for I=1 to 66;A[I,1]→A[I,3];next I
75: gto "START"
76:
77:
78: "BREAKOUT":
79: "Decode byte string Z into proper slots in A":
80: "PRESS ":(256Z[3]+Z[2])/20)→A[15,1];if bit(0,Z[10]);-A[15,1]→A[15,1]
81: "PRESS CORR":.9989789716A[15,1]+.30325283→A[15,1]
82: "TEMP-S":(256Z[5]+Z[4])/2000→A[16,1]
83: "COND":(256Z[7]+Z[6])/1000→A[18,1]
84: "F-TEMP":(256Z[9]-Z[8])/2000→A[17,1]
85: "SPECIAL FIX FOR GYRE CRUISE ONLY":A[17,1]→A[16,1]
86: for I=11 to 27 by 2;shf(Z[I+1],-6)+shf(Z[I],2)→T[I]
87: if bit(0,Z[I])=1;-T[I]→T[I]
 : next I
 : "VELOC ":for I=1 to 3;T[11+2(I-1)].00639→A[22-I,1];next I
90: -A[20,1]→A[20,1]
91: "MAGNET":for I=1 to 3;T[17+2(I-1)]/1000→A[21+I,1];next I
92: -A[22,1]→A[22,1]
93: "ACCEL ":for I=1 to 3;T[23+2(I-1)]/1000→A[24+I,1];next I
94: -A[26,1]→A[26,1]
95: "TIME":
96: shf(Z[38],4)→T[1];Z[38]-shf(T[1],-4)→T[2]
97: shf(Z[37],4)→T[3];Z[37]-shf(T[3],-4)→T[4]
98: shf(Z[36],4)→T[5];Z[36]-shf(T[5],-4)→T[6]
99: shf(Z[35],4)→T[7];Z[35]-shf(T[7],-4)→T[8]
100: shf(Z[34],4)→T[9];Z[34]-shf(T[9],-4)→T[10]
101: shf(Z[33],4)→T[11];Z[33]-shf(T[11],-4)→T[12]
102: 100T[1]+10T[2]+T[3]→T[13]
103: 10T[4]+T[5]→T[14]
104: 10T[6]+T[7]→T[15]
105: 10T[8]+T[9]+T[10]/10+T[11]/100+T[12]/1000→T[16]
106: for I=1 to 4;T[I+12]→A[1+I,1];next I
107: ret
108:
109: "DERIVE":
110:
111: 'SALINITY'(A[15,1],A[16,1],A[18,1])→A[28,1]
112: cll 'SIGMA-T'(A[16,1],A[28,1],A[30,1])
113: cll 'SND SPEED'
114: cll 'ANGLE'
115: cll 'TRANSFORMS'
116: cll 'TRANSFORM Q'
117: cll 'GMAG CURRENT'
118: cll 'GMAG ACCEL'
119: cll 'GMAG MAG'
120: cll 'INST VEL'
121: cll 'BEST VEL'
122: if abs(A[15,1]-A[15,3])<1;ret
123: cll 'dp/dz'
124: cll 'VEL SHEARS'
125: cll 'N AND RI'
126: ret
127:
128: "SALINITY":
129: "ARGUMENTS: 1-A[15,1],2-A[16,1],3-A[18,1]":
130: 1.45038/9.9e7→p4
131: 6.76583621732e5→p5;2.00529363371e2→p6
132: 1.11098951612e-2→p7;-7.26681983149e-7→p8
133: 1.3586827285e-11→p9
134: p3*(1-5.25e-6*(p2-15)+p1p4)/42.906→p11
135: 1.60836e-5p1-5.4845e-10p1↑2+6.166e-15p1↑3→p12
```

```
p12/(1+.030786p2+3.169e-4p2↑2)→p13
p11/(1+p13)→p14
100p2→p10;(p5+p6p10+p7p10↑2+p8p10↑3+p9p10↑4)(1*tn↑(-6))→p15;p14/p15→p16
-.08996+28.8567p16+12.18882p16↑2-10.61869p16↑3→p17
p17+5.98624p16↑4-1.32311p16↑5→p17
p17+p16(p16-1)(.0442p2-.00046p2↑2-.004p16p2)→p18
ret p18


"SIGMA-T":
"ARGUMENTS:1-A[16,1],2-A[28,1]":
"VARIABLES DESTROYED : C[*],I,J":
if not flg10;dim C[0:4,0:4];sfg 10
for I=0 to 3
for J=0 to 3
0→C[I,J]
next J
next I
8.00969062e-2→C[0,0];7.97018644e-1→C[0,1]
1.31710842e-4→C[0,2];-6.11831499e-8→C[0,3]
5.8819403e-2→C[1,0];-3.25310441e-3→C[1,1];2.8797153e-6→C[1,2]
-8.11465413e-3→C[2,0];3.89187483e-5→C[2,1]
4.76600414e-5→C[3,0]
0→p3
for I=0 to 3
for J=0 to 3
if I+J<4;p3+C[I,J]*p1↑I*p2↑J→p3
next J
next I
ret

"SND SPEED":
"SOUND SPEED FORMULA":
A[16,1]→X;A[28,1]→S;A[15,1]→D
100(1449+4.6X-.055XX+.0003X↑3+(1.39-.012X)(S-35)+.017D)→C
C→A[29,1]
ret

"TRANSFORMS":A[49,1]→p1
deg;cos(p1)→p2;-sin(p1)→p3
0→X→Y
for I=1 to 3;A[I+21,1]↑2+X→X;A[I+24,1]↑2+Y→Y
next I
√X→X;√Y→Y
for I=1 to 3;A[I+21,1]/X→E[I];A[I+24,1]/Y→E[I+3]
next I
for I=1 to 3;-E[I+3]→B[3,I];next I
for I=1 to 3;(E[I]-p3B[3,I])/p2→B[2,I];next I
sgn(A[22,1])√(1-B[2,1]↑2-B[3,1]↑2)→B[1,1]
A[26,1]A[24,1]-A[27,1]A[23,1]→p1
-B[1,1](A[25,1]A[24,1]-A[27,1]A[22,1])/p1→B[1,2]
B[1,1](A[25,1]A[23,1]-A[26,1]A[22,1])/p1→B[1,3]
ret

"TRANSFORM Q":
for J=1 to 3;F[1,J]→X
for I=1 to 3;A[I+18,1]F[I+1,J]+X→X;next I
X→A[J+30,1];next J
ret

"GMAG CURRENT":
"CURRENTS IN GEOMAG. COORD. FROM INSTRUMENT COORD.":
for I=1 to 3
0→Y
for J=1 to 3;B[I,J]A[J+30,1]+Y→Y;next J
Y→A[I+39,1]
next I
ret

"GMAG ACCEL".
"ACCELERATIONS, LESS G, IN GEOMAGNETIC COORDINATES":
```

```
for I=1 to 3;0→X
for J=1 to 3;B[I,J]A[J+24,1]+X→X;next J
X→A[I+33,1];next I
A[36,1]+9.9→A[36,1]
ret

"GMAG MAG":
"MAGNETIC FIELD IN GEOMAGNETIC COORDINATES":
for I=1 to 3;0→Y
for J=1 to 3;B[I,J]A[J+21,1]+Y→Y;next J
Y→A[I+36,1];next I
ret

"ANGLE":
"COMPUTATION OF ANGLE BET. G' AND H'":
√(A[22,1]↑2+A[23,1]↑2+A[24,1]↑2)→H→A[50,1]
A[22,1]/H→D[1];A[23,1]/H→D[2];A[24,1]/H→D[3]
√(A[25,1]↑2+A[26,1]↑2+A[27,1]↑2)→G→A[51,1]
A[25,1]/G→D[4];A[26,1]/G→D[5];A[27,1]/G→D[6]
0→B;for J=1 to 3;D[J]D[J+3]+B→B;next J
acn(B)→B;B→A[49,1]→A[9,1]
ret

"INST VEL":
"INSTRUMENT VELOCITY":
A[2,1]→p1
(A[2,1]-p1)86400+3600A[3,1]+60A[4,1]+A[5,1]→p3
(A[2,2]-p1)86400+3600A[3,2]+60A[4,2]+A[5,2]→p2
if flg7;p3-p2→A[55,1]
if A[55,1]=0;5→A[55,1]
if not flg7;5→A[55,1];sfg 7
100(A[15,1]-A[15,2])/A[55,1]→A[45,1]
0→A[43,1]→A[44,1]
ret

"BEST VEL":
"CORRECTS OBSERVED VELOCITIES FOR INSTRUMENT MOTION":
for I=1 to 3;A[I+39,1]-A[I+42,1]→A[I+45,1];next I
ret

"dp/dz":
"VERTICAL DENSITY GRADIENT (dp/dz)":
A[15,1]-A[15,3]→A[56,1]
(A[30,1]-A[30,3])/A[56,1]→A[57,1]
ret

"VEL SHEARS":
"VERTICAL VELOCITY SHEARS ":
for I=46 to 48;(A[I,1]-A[I,3])/A[56,1]→A[I+12,1]
ret

"N AND RI":
"N AND RICHARDSON NUMBER":
980A[57,1]/(1+A[30,1]/1000)→p1;if p1<0;-p1→p1
√p1→A[62,1]
A[58,1]↑2+A[59,1]↑2→p2
if p2≠0;p1/p2→A[61,1]
if p2=0;999→A[61,1]
ret

"DISPLAY":
cll 'HAZELTINE'
cll 'TIMESERIES'
cll 'DEPTH'
cll 'DEPTH2'
cll 'V1V2'
cll 'STATS'
ret

"HAZELTINE":if O[1]<0;abs(O[1])→O[1];cll 'HAZINITIALIZE'
if O[1]=0;ret
```

```
278: cll 'HAZWRITE'
279: ret
280: "HAZINITIALIZE":
281: if not flg5;dim L$[3,80];sfg 5
282: " "→L$[1]
283: for I=1 to V[O[1],0];V$[V[O[1],I],1,6]→L$[1,1+6(I-1),6]];next I
284: wtb 3,27,26;wtb 3,27,17,0,3
285: fmt 1,c80,z;wrt 3,L$[1];ret
286: "HAZWRITE":
287: fmt 1,f6.2,z;fmt 2,/
288: wtb 3,27,17,0,5;wtb 3,27,26
289: for I=1 to V[O[1],0];wrt 3.1,A[V[O[1],I],1];next I
290: wrt 3.2;wtb 3,27,17,40,2
291: fmt 3,f3.0,2x,f2.0,2x,f2.0,2x,f6.3
292: wrt 3.3,A[2,1],A[3,1],A[4,1],A[5,1]
293: wtb 3,27,17,2,2
294: fmt 4,"CAST # ",f4.0;wrt 3.4,A[1,1]
295: wtb 3,27,17,2,1;fmt 5,"LATITUDE  ",f6.3,"    LONGITUDE  ",f8.3
296: wrt 3.5,A[6,1],A[7,1]
297: ret
298:
299: "DEPTH":if O[3]<0;-O[3]→O[3];0→O[4];cll 'DEPTHINITIALIZE'
300: if O[3]=0;ret
301: cll 'DEPTHPLOT'
302: ret
303: "DEPTHINITIALIZE":wtb 706,27,85
304: wrt 705,"IP,1000,1000,6000,6000";psc 705;pclr
305: fxd 0
306: csiz 3.5;scl 0,10,0,10;plt 0,16,1;lbl T$
307: csiz 2;scl 0,1,300,0;xax 0;xax 300;yax 1,20;yax 0,20,0,300,5
308: csiz 3,2,1,90;plt -.1,170,1;lbl "Pressure";csiz 3,2,1,0
309: for I=1 to V[O[3],0]
310: val(V$[V[O[3],I],7,13])→p1
311: val(V$[V[O[3],I],14,20])→p2
312: A[15,1]→R[0];A[V[O[3],I],1]→R[I]
313: 5→p6;if abs(p2-p1)<10;1→p6
314: 1→p7;if abs(p2-p1)>99;50→p7;1→p6
315: csiz 2;scl p1,p2,0,10;xax 10+I,p7,p1,p2,p6
316: csiz 3;scl 0,80,0,10;plt 30,10.1+I,1;lbl V$[V[O[3],I],1,6]
317: csiz 1.5
318: next I
319: ret
320: "DEPTHPLOT":psc 705
321: for I=1 to V[O[3],0]
322: val(V$[V[O[3],I],7,13])→p1
323: val(V$[V[O[3],I],14,20])→p2
324: scl p1,p2,300,0;lim p1,p2,300,0
325: A[V[O[3],I],1]→p3;A[15,1]→p4
326: plt p3,p4,1;plt p3,p4,2
327: p3→R[I];p4→R[0]
328: next I
329: lim
330: ret
331:
332: "STATS":if O[4]=0;ret
333: if O[4]<0;-O[4]→O[4];0→O[3];cll 'STATSINITIALIZE'
334: cll 'STATSXEQ'
335: ret
336: "STATSINITIALIZE":wtb 706,27,85
337: wrt 706,T$
338: for I=1 to 7;0→S[I];next I
339: fmt 1,20x,"INSTRUMENT ATTITUDE STATISTICS",/
340: fmt 2,5x,"PITCH",14x,"YAW",13x,"MAG DIP",13x,"PRESSURE"
341: fmt 3," Mean   Std Dev    Mean    Std Dev    Mean    Std Dev    Mean",/
342: wrt 706.1;wrt 706.2;wrt 706.3
343: ret
344: "STATSXEQ":S[1]+1→S[1]
345: "Pitch":deg;acs(B[3,3])→X;cll 'SUPDATE1'(2,X)
346: "Yaw":acs(B[1,1])→X;cll 'SUPDATE1'(4,X)
347: "MagDip":A[49,1]→X;cll 'SUPDATE1'(6,X)
348: "Press ":A[19,1]→X;cll 'SUPDATE1'(8,X)
```

```
if S[1]=20;gto "SLIST"
ret
"SLIST":for I=2 to 8 by 2;gsb "SUPDATE2"
next I
fmt 1,f7.2,2x,z;fmt 2,/
for I=2 to 8;wrt 706.1,S[I];next I;wrt 706
for I=1 to 7;0→S[I];next I
ret
"SUPDATE1":S[p1]+p2→S[p1];S[p1+1]+p2↑2→S[p1+1];ret
"SUPDATE2":S[I]/S[1]→S[I];√(S[I+1]/S[1]-S[I]↑2)→S[I+1];ret

"TIMESERIES":if O[2]<0;-O[2]→O[2];0→O[5];cll 'TIMESERIESINITIALIZE'
if O[2]=0;ret
cll 'TIMESERIESPLOT'
ret
"TIMESERIESINITIALIZE":wtb 704,27,85
fxd 0;cfg 8
psc 703;pclr
wrt 703,"IP,1000,1000,7500,10000"
csiz 2
scl 0,2000,0,1;xax 0,100,0,2000,5;yax 2000;xax 1;yax 0
V[O[2],0]→p1
for I=1 to p1-1;xax I/p1,100;next I
csiz 2,2,1,90
for I=1 to p1;plt -200,(I-1)/p1+.02,1;lbl V$[V[O[2],I],1,6];next I
9000/p1→p2
for I=1 to p1;fmt 1,"IP",",",fz4.0,",",fz4.0,",",fz5.0,",",fz5.0
wrt 703.1,1000,1000+(I-1)p2,7500,1000+Ip2
csiz p1,2,1/6.5,0
val(V$[V[O[2],I],7,13])→p3
val(V$[V[O[2],I],14,20])→p4
scl 0,2000,p3,p4;fxd 0;5→p6;if abs(p4-p3)<10;1→p6
1→p7;if abs(p4-p3)>50;2→p6;50→p7
if Imod2;yax 0,p7,p3,p4,p6
if not Imod2;yax -80,p7,p3,p4,p6
next I
ret
"TIMESERIESPLOT":psc 703
if not flg6 or flg8;gto "TS1next"
A[3,1]→Q[3];A[4,1]→Q[4];A[5,1]→Q[5]
A[2,1]→Q[1];3600A[3,1]+60A[4,1]+A[5,1]→Q[2];sfg 8
wrt 703,"IP,1000,1000,7500,10000";scl 0,80,0,10
fxd 0;csiz 1.5
plt 10,10.3,1;lbl "Start Time ",Q[3]," :",Q[4]," :",Q[5]
plt 10,10.5,1;lbl "Start Day ",Q[1]
"TS1next":
if flg6;gto "TSnext"
dim Q[7];A[2,1]→Q[1];3600A[3,1]+60A[4,1]+A[5,1]→Q[2]
sfg 6
"TSnext":
0→p8;if A[2,1]>Q[1];86400→p8
V[O[2],0]→p1;9000/p1→p2
p8+3600A[3,1]+60A[4,1]+A[5,1]-Q[2]→p9
for I=1 to p1;fmt 1,"IP",",",fz4.0,",",fz4.0,",",fz5.0,",",fz5.0
wrt 703.1,1000,1000+(I-1)p2,7500,1000+Ip2
val(V$[V[O[2],I],7,13])→p3
val(V$[V[O[2],I],14,20])→p4
scl 0,2000,p3,p4;lim 0,2000,p3,p4
A[V[O[2],I],1]→p10
plt p9,p10,1
plt p9,p10,2
next I
lim
ret
"ERRORHANDLER":if ern=4;prt "TIMEOUT ERROR"
prt "ERN =",ern
prt "Line",erl
prt "ROM",rom
time 0
cli 7
cli 5
```

```
420: gto "START"
421:
422: "OPTIONS":if A=0;ret
423: ent "Enter Display Title",T$
424: "YESYES":ent "Enter Display Device No.",A;if A=0;ret
425: if A>9 or A<0;jmp -1
426: ent "Enter display option no.",O[A];if O[A]>0;-O[A]→O[A]
427: ent "Do you want more?",Q$;if cap(Q$[1,1])="Y";gto "YESYES"
428: if cap(Q$[1,1])#"Y" and cap(Q$[1,1])#"N";jmp -1
429: 0→A;ret
430:
431: "SIGTEST":fmt 1,3f10.5
432: ent "enter T",T;ent "enter S",S
433: cll 'SIGMA-T'(T,S,Q)
434: wrt 3.1,T,S,Q
435: gto "SIGTEST"
436: "TSTST":cll 'TIMESERIESPLOT'
437: stp
438: "DEPTH2":if O[5]<0;-O[5]→O[5];0→O[2];cll 'DEPTHINITIALIZE2'
439: if O[5]=0;ret
440: cll 'DEPTHPLOT2'
441: ret
442: "DEPTHINITIALIZE2":wtb 704,27,85
443: wrt 703,"IP,1000,1000,6000,6000";psc 703;pclr
444: fxd 0
445: csiz 3.5;scl 0,10,0,10;plt 0,16,1;lbl T$
446: csiz 2;scl 0,1,300,0;xax 0;xax 300;yax 1,20;yax 0,20,0,300,5
447: csiz 3,2,1,90;plt -.1,170,1;lbl "Pressure";csiz 3,2,1,0
448: for I=1 to V[O[5],0]
449: val(V$[V[O[5],I],7,13])→p1
450: val(V$[V[O[5],I],14,20])→p2
451: A[15,1]→R[0];A[V[O[5],I],1]→R[I]
452: 5→p6;if abs(p2-p1)<10;1→p6
453: 1→p7;if abs(p2-p1)>99;50→p7;1→p6
454: csiz 2;scl p1,p2,0,10;xax 10+I,p7,p1,p2,p6
455: csiz 3;scl 0,80,0,10;plt 30,10.1+I,1;lbl V$[V[O[5],I],1,6]
456: csiz 1.5
457: next I
458: ret
459: "DEPTHPLOT2":psc 703
460: for I=1 to V[O[5],0]
461: val(V$[V[O[5],I],7,13])→p1
462: val(V$[V[O[5],I],14,20])→p2
463: scl p1,p2,300,0;lim p1,p2,300,0
464: A[V[O[5],I],1]→p3;A[15,1]→p4
465: plt p3,p4,1;plt p3,p4,2
466: p3→R[I];p4→R[0]
467: next I
468: lim
469: ret
470: "V1V2":if O[6]<0;-O[6]→O[6];cll 'V1V2INIT'
471: if O[6]=0;ret
472: cll 'V1V2PLOT'
473: ret
474: "V1V2INIT":ent "Enter the plot unit",Z
475: if Z=703;0→O[5]→O[2]
476: if Z=705;0→O[4]→O[3]
477: wtb Z+1,27,85
478: fxd 0;psc Z
479: pclr
480: wrt Z,"IP,1000,1000,6000,6000"
481: ent "Variable number 1",U[1]
482: ent "Variable number 2",U[2]
483: csiz 2
484: val(V$[U[1],7,13])→p1;val(V$[U[1],14,20])→p2
485: val(V$[U[2],7,13])→p3;val(V$[U[2],14,20])→p4
486: scl p1,p2,p3,p4;5→p6;if abs(p2-p1)<10;1→p6
487: 1→p7;if abs(p2-p1)>99;50→p7;1→p6
488: xax p3,p7,p1,p2,p6;xax p4
489: 5→p6;if abs(p4-p3)<10;1→p6
490: 1→p7;if abs(p4-p3)>99;50→p7;1→p6
```

```
491: yax p1,p7,p3,p4,p6;yax p2
492: csiz 3;scl 0,80,0,10;plt 30,11.1,1;lbl V$[U[1],1,6]
493: csiz 3.5;scl 0,10,0,10;plt 0,15,1;lbl T$
494: scl 0,10,0,80;csiz 3,2,1,90;plt -1,30,1;lbl V$[U[2],1,6]
495: csiz ;lim
496: ret
497: "V1V2PLOT":psc Z
498: val(V$[U[1],7,13])→p1;val(V$[U[1],14,20])→p2
499: val(V$[U[2],7,13])→p3;val(V$[U[2],14,20])→p4
500: scl p1,p2,p3,p4;lim p1,p2,p3,p4
501: A[U[1],1]→p5;A[U[2],1]→p6
502: plt p5,p6,1;plt p5,p6,2;lim
503: ret
*9743
```

VARIABLES IN DISPLAY OPTIONS

OPTION #	1
1	DAY
2	HR
3	MIN
4	SEC
5	CAST
6	LAT
7	LONG

OPTION #	2
1	PRESS
2	U1-GC
3	U2-GC
4	U3-GC
5	UI3-GC
6	ABSG-0
7	ABSH-0
8	MAGDIP

OPTION #	3
1	T-SLOW
2	PRESS
3	A1-GC
4	A2-GC
5	A3-GC
6	H1-GC
7	H2-GC
8	H3-GC

OPTION #	4
1	PRESS
2	U1-IC
3	U2-IC
4	U3-IC
5	Q1-AP
6	Q2-AP
7	Q3-AP
9	THETA
9	PHI

OPTION #	5
1	PRESS
2	U1-GC
3	U2-GC
4	U3-GC
5	ABSG-0
6	N
7	RI
8	T-SLOW
9	SAL

OPTION #	6
1	T-SLOW
2	SAL
3	SIGMAT

OPTION #	7
1	G1-IC
2	G2-IC
3	G3-IC
4	H1-IC
5	H2-IC
6	H3-IC

OPTION #	8
1	U1-GC
2	U2-GC

```
;A*LIB(1).TRANSCRIBE
C******************************************************************
C******************************************************************
C*** PROGRAM : TRANSCRIBE                                       ***
C*** PURPOSE : TRANSLATES PROFILER DATA TO ENGINEERING UNITS    ***
C***           LEAVING VELOCITY DATA IN NON-ORTHOGONAL          ***
C***           INSTRUMENT COORDINATES.                          ***
C***                                                            ***
C*** AUTHOR  : J.J. VEGA , COMPUTER SCIENCES CORP. (PRIMARY)    ***
C***           K.D. SAUNDERS , NORDA ( SECONDARY - REQUIRED IN  ***
C***                  ORDER TO CORRECT VEGA'S CODE AND COMPLETE ***
C***                  DOCUMENTATION.)                           ***
C***                                                            ***
C*** DATE (OF LATEST REVISION) : 15 MARCH 1982                  ***
C***                                                            ***
C*** INPUT : FILE           TYPE OF DATA/COMMENTS               ***
C***          5             INPUT FROM CARDS/TERMINAL           ***
C***                        (IF USED FROM A TERMINAL, THE PROGRAM ***
C***                        SUPPLIES SOLICITATION PROMPTS )     ***
C***         10             DATA TAPE IN HP 9825A / NORDA FORMAT ***
C***                        ( THE TAPE SHOULD BE COPIED FROM THE ***
C***                        800 BPI, RAW TAPE TO A HIGHER DENSITY ***
C***                        TAPE FOR TWO REASONS: FIRST, THE    ***
C***                        PROGRAM WILL RUN FASTER AND, SECOND, ***
C***                        YOU WILL NOT GET ABNORMAL FRAME COUNTS ***
C***                        WHICH WILL TERMINATE THE PROGRAM! THE ***
C***                        PROPER METHOD FOR COPYING IS USE    ***
C***                        @COPY,MN   INPUT,10 .)              ***
C***                                                            ***
C*** OUTPUT                                                     ***
C***                                                            ***
C***         20             FEB FILE CONTAINING PROFILER DATA.  ***
C***                        ( THIS FILE MUST BE ASSIGNED PRIOR TO ***
C***                        EXECUTING THE PROGRAM.)             ***
C***                                                            ***
C******************************************************************
C******************************************************************
C
C
C   MAIN CALLING ROUTINE FOR TRANSCRIBE ROUTINE
C
C   THIS PROGRAM READS DATA FROM MAG TAPE,
C   DECODES IT, AND STORES IT IN FEB FILES.
C
C
C   ARRAY LIST
C       DBLK   IS THE INPUT DATA ARRAY
C              READ FROM TAPE.
C       RD     IS THE ARRAY CONTAINING
C              THE BYTES OF THE DATA
C              STRING TO BE DECODED.
C       VAR    IS THE ARRAY CONTAINING
C              THE DECODED VARIABLES OF
C              A DATA STRING.
C       WW     IS THE FEB FILE DATA ARRAY
C       ADOCW  IS THE FEB FILE ALPHA-
C              NUMERIC HEADER ARRAY.
C       FDOCW  IS THE FEB FILE FLOATING
C              POINT HEADER ARRAY.
C       IDOCW  IS THE FEB FILE INTEGER
C              HEADER ARRAY.
C
C   SEE NAVO TECH NOTE 'THE FAST AND
```

```
C         EASY BINARY (FEB) FILE' BY
C         Z.R. HALLOCK FOR MORE ON FEB
C         FILES.
C
C    COMMONS FOR ZWRIT
C
          COMMON/WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(15)
          COMMON/WDATA/VW(15,500)
          COMMON/WDOCF/FDOCW(40)
          COMMON/WDOCI/IDOCW(20)
          COMMON/WDOCA/ADOCW(50)
C
          COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
C    INITIALIZE CONTROL HEADER
C
          DATA LW,NW,NFW,NIW,NAW/15,500,40,20,5(
          DATA (IPW(I),I=1,15)/'PRESS','STEMP','COND','FTEMP',
     $    'VLOCI1','VLOCI2','VLOCI3','MAGI1','MAGI2','MAGI3','ACCLI1',
     $    'ACCLI2','ACCLI3','TIME','RELSEC'/
          DATA NNNW,NNIP,NNF,NNI,NNA/500,15,40,20,50/
C
C    READ ADOC ARRAY
             WRITE(6,5000)
          READ (5,10) (ADOCW(I),I=1,7)
          READ (5,10) (ADOCW(I),I=13,19)
          READ (5,10) (ADOCW(I),I=25,31)
C
C    READ IDOC ARRAY
             WRITE(6,5001)
          READ (5,40) IDOCW(2),IDOCW(3),IDOCW(6),IDOCW(10),IDOCW(14)
C
C    READ FDOC ARRAY
             WRITE(6,5002)
          READ (5,40) (FDOCW(I),I=1,9)
             WRITE(6,5003)
          READ (5,40) (FDOCW(I),I=16,19)
             WRITE(6,5004)
          READ (5,40) (FDOCW(I),I=21,27)
             WRITE(6,5005)
          READ (5,40) (FDOCW(I),I=31,39)
C
C    READ FILE NAME, SEGMENT NAME, MSG LEVEL.
             WRITE(6,5006)
          READ (5,50) NMFW
             WRITE(6,5007)
             READ(5,50) NMBW
             WRITE(6,5008)
          READ (5,40) MSGW
C
C
10        FORMAT (7A6)
40        FORMAT ()
50        FORMAT (A6)
C
          ADOCW(37)=' INST.'
          ADOCW(38)=' COORD'
          ADOCW(39)='      '
C    CALL SUBROUTINE TRANSC
          CALL TRANSC
C
C
5000        FORMAT(' ENTER 3 LINES OF ALPHAMERIC DOCUMENTATION')
```

```
      5001   FORMAT(' ENTER AS INTEGERS:CRUISE NO.,STATION NO.,',
     1 'ABSOLUTE NO. OF 1ST SEG.,INPUT TAPE NO.,YEAR OF CAST')
      5002   FORMAT(' ENTER (FLOATING PT.) : '/
     1 ' SAMPLE INTERVAL (SEC), ZERO, STARTING LATITUDE (DEC. DEG.)'/
     2 ' STARTING LONGITUDE, TIME OF FIX AT START (DEC. DAYS)'/
     3 ' ENDING LATITUDE, ENDING LONGITUDE, ENDING TIME,'/
     4 ' MAXIMUM PRESSURE (DECIBARS) '//)
      5003   FORMAT( ' ENTER (F) : MAGNETIC VARIATION AND DIP,'/
     1 ' SHIP SPEED AND HEADING '//)
      5004   FORMAT(' ENTER DRY BULB TEMP., WET BULB TEMP.,SURFACE TEMP.'
     1 /' BAROMETRIC PRESSURE, WIND SPEED (KTS), WIND DIRECTION,'/
     2  ' SIGNIFICANT WAVE HEIGHT (FT) '//)
      5005   FORMAT(' ENTER (F) : CAST START DAY, HOUR, MIN,SEC'/
     1 ' ZERO, CAST END TIME: DAY, HOUR, MIN, SEC'//)
      5006   FORMAT(' ENTER THE SEQUENTIAL FILE NUMBER (I) ')
      5007   FORMAT(' ENTER NMEW WHICH CONSISTS OF A 3 DIGIT CAST NO.,'
     1 ' AND A 3 DIGIT PROFILE NO.'//)
      5008   FORMAT(' ENTER THE MESSAGE LEVEL 0-9'//)
             END
```

```
@PRT,S V.TRANSC

@A*LIB(1).TRANSC
             SUBROUTINE TRANSC
      C
      C  THIS ROUTINE READS SINGLE DATA BLOCKS FROM
      C  MAG TAPE AND CALLS SUBROUTINES TO FIND AND
      C  DECODE EACH DATA STRING.  AFTER 500 DATA
      C  CYCLES ARE DECODED, A FEB SEGMENT IS WRITTEN.
      C
      C  COMMONS FOR ZWRIT
      C
             COMMON/WHDR/LW,NW,NBW,NNBW,NMFW,NFW,NIW,NAW,IPW(15)
             COMMON/WDATA/VW(15,500)
             COMMON/WDOCF/FDOCW(40)
             COMMON/WDOCI/IDOCW(20)
             COMMON/WDOCA/ADOCW(50)
      C
             COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
             COMMON/EETA/BMAT(3,3),ORTHOG(4,3),ACCAL(3)
      C  INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
             DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
          $ -.055,-.803,1.58,-.794/
             DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
          $ -.197,-.684,-.011,-.759/
             DATA  ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
          $ .612,.849,-.071,.889/
      C
                 INTEGER BLKSIZ
                 DATA BLKSIZ/495/
             DATA I2,J,K,MFLG,NFLG,IB/6*0/
             DIMENSION VAR(17),RD(37)
                 INTEGER DBLK(500)
             DOUBLE PRECISION DPTIME,RELDAY
                   REAL W(3),LENGTH
             KFLG=1
             IDOCW(1)=0
             IDOCV(4)=0
             NMBYTE=1
             N2=0
             IFIRST=1
```

```
            NBAD=0
               ISEG = 0
               NBDTOT = 0
      C
      C  *****
      C
   10       CONTINUE
      C
            IF (KFLG.EQ.0) GO TO 30
      C
      C  *****
      C  READ NEXT INPUT BLOCK.
   20         CONTINUE
               DO 1000 I = 1 ,600
               DBLK(I) = 0
 1000          CONTINUE
                  KFLG = 0
            CALL NTRAN (10,2,414,DBLK,L)
            IF (L.EQ.-1) CALL NTRAN (10,22)
            IF (L.EQ.-2) GO TO 22
            IF (L.EQ.-3.OR.L.EQ.-4) GO TO 1020
            KFLG=0
            NMBYTE=1
            GO TO 30
      C
      C  *****
      C  IF EOF READ, WRITE PARTIAL SEGMENT AND TERMINATE ROUTINE.
   22       NFLG=1
            IDOCW(1)=1
            WRITE (6,23)
   23       FORMAT (' EOF READ BEFORE SPECIFIED END OF CAST TIME',/,
         $  ' ** PARTIAL SEGMENT WRITTEN **')
      C
      C  *****
      C  CALL ROUTINE TO READ AND DECODE DATASTRING
   30       CONTINUE
            CALL DATSTR (KFLG,DBLK,BLKSIZ,NMBYTE,RD,VAR)
      C
            IF (KFLG.EQ.1 .AND. NFLG .NE. 1) GO TO 20
      C
      C  *****
      C
      C
      C  DECODED DATA TO FEB DATA ARRAY
      C
      C  CHECK FOR BAD CYCLE
      C
      C
      C*****************************************************************
      C                 CHECK THAT THE PRESSURE IS IN RANGE            *
      C*****************************************************************
      C
            IF (VAR(1).GT.FDOCW(9).OR.
         $     VAR(1).LT.-5) GO TO 999
      C
      C*****************************************************************
      C                 CHECK THAT THE DATE IS IN RANGE                *
      C*****************************************************************
      C
            IF (VAR(14).GT.FDOCW(36).OR.
         $     VAR(14).LT.FDOCW(31)) GO TO 999
      C
```

```
C******************************************************************
C      CHECK THAT THE TIME FIELDS ARE EACH IN RANGE               *
C******************************************************************
C
          IF (VAR(15).GE.24.000) GO TO 999
          IF (VAR(16).GE.60.000) GO TO 999
          IF (VAR(17).GE.60.000) GO TO 999
          IF (IFIRST.NE.1) GO TO 75
C
C  COMPUTE DPTIME FOR FIRST CYCLE OF FIRST SEGMENT ONLY.
          DPTIME=VAR(14)+VAR(15)/24.+VAR(16)/1440.+VAR(17)/86400.
          IFIRST=0
C
75        CONTINUE
C
C  COMPUTE RELSEC FOR ALL CYCLES.
          RELDAY=VAR(14)+VAR(15)/24.+VAR(16)/1440.+VAR(17)/86400.
          RELSEC=86400.*(RELDAY-DPTIME)
             N2 = N2 + 1
C  ORTHOGONALIZE VELOCITY VECTOR
C
          DO 9910 J=1,3
          XO=OPTHOG(1,J)
          DO 5 I=1,3
5         XO=VAR(I+4)*ORTHOG(I+1,J)+XO
9910         W(J)=XO
         DO 60 I=1,3
60       VAR(I+4)=W(I)
               LENGTH = SQRT(W(1)*W(1)+W(2)*W(2)+W(3)*W(3))
               WRITE(8,1999) LENGTH
1999           FORMAT( ' LENGTH OF ABSOLUTE VELOCITY PRIOR TO TRANSFORM
        1      ,G20.5)
          WRITE(8,2999) W(1),W(2),W(3)
2999         FORMAT(' V - BEFORE :', 3G20.7)
          DO 150 I=1,13
          I3=I
150       VW(I,N2)=VAR(I)
          VW(14,N2)=DPTIME
          VW(15,N2)=RELSEC
C
          IF (N2.LT.500.AND.NFLG.NE.1) GO TO 10
             IF( N2.LT.500 .AND. NFLG .EQ. 1 .AND. KFLG .EQ. 0)GOTO30
C
C  *****
C  WRITE FEB SEGMENT
C
40           IB=0
          IF (NFLG.EQ.1) NW=N2
          N2=0
          IDOCW(4)=IDOCW(4)+1
          IDOCW(13)=NBAD
          FDOCW(10)=DPTIME
          FDOCW(13)=FDOCW(36)+FDOCW(37)/24.+FDOCW(38)/1440.
     $             +FDOCW(39)/86400.
          CALL ZWRIT (20,IF,IB)
             ISEG = ISEG + 1
             IF(NFLG .EQ. 1) RETURN
          NBAD=0
C
C  CLEAR DATA ARRAY
C
          DO 200 I=1,15
          DO 200 J=1,500
```

```
      200     VW(I,J)=0.00
      C
      999     CONTINUE
      C  COUNT AND OMIT BAD CYCLES.
              NBAD=NBAD+1
                NBDTOT = NBDTOT + 1
      C
      C***************************************************************
      C  WRITE INFORMATION CONCERNING THE BAD RECORD TO UNIT 9        *
      C***************************************************************
      C
      C
                WRITE(9,8000) N2,ISEG,NBDTOT,VAR(1),VAR(14),VAR(15),VAR(16)
      8000      FORMAT( 3I5,4G16.5)
      C  *****
      C  TEST FOR END OF CAST
      C
                 IF(NFLG .EQ. 1 .AND. KFLG .EQ. 0) GOTO30
                 IF(NFLG.EQ.1 .AND. KFLG .EQ. 1) GOTO40
              IF (NFLG.NE.1) GO TO 10
      C  IF END OF CAST, RETURN TO CALLING ROUTINE
      C  AND TERMINATE FEB FILE.
      C
      C  *****
              RETURN
      1020    WRITE (6,1021) L
      1021    FORMAT (' L=',I3,' IN DATABLOCK READ.')
              END
```

```
@PRT,S V.DATASTRING

SA*LIB(1).DATASTRING
             SUBROUTINE DATSTR (KFLG,INBUF,IBUFSZ,NMBYTE,RD,VAR)
    C
    C  THIS ROUTINE SCANS THE ARRAY DBLK, BY BYTE, FOR
    C  A VALID FRAME SYNC NUMBER.  WHEN ONE IS FOUND,
    C  EACH BYTE OF THE DATA STRING IS TRANSFERED TO
    C  A SEPARATE ELEMENT OF ARRAY RD, AND SUBROUTINE
    C  DECODE IS CALLED TO DECODE THE STRING.
    C
             DIMENSION IFS(2),RD(37),VAR(17)
               INTEGER FS,FS2,FS3
    C
    C *****
    15       CONTINUE
    C
    C *****
    C  SCAN FOR F.S.
             CALL MOVE (INBUF,NMBYTE,IFS(1),1,1)
             NMBYTE=NMBYTE+1
                IF(NMBYTE .LE. 534) GOTO30
               NMBYTE = 1
                 KFLG = 1
                 RETURN
    30        FS=FLD(0,8,IFS(1))
             IF (FS.NE.15.AND.FS.NE.240) GO TO 15
               NPLUS = NMBYTE + 26
               CALL MOVE(INBUF,NPLUS,IFS(1),1,1)
               FS2 = FLD(0,8,IFS(1))
               NPLUS = NMBYTE+8
               CALL MOVE(INBUF,NPLUS,IFS(1),1,1)
               FS3 = 0
```

```
      FS3 = FLD(0,5,IFS(1))
      IF(FS2 .NE. 255) GOTO 15
      IF(FS3.NE.31) WRITE(16,8777) IFS(1),FS3
 8777 FORMAT(5X,O12,I10)
      IF(FS3 .NE. 31) GOTO 15
C *****
C
C *****
C   MOVE NEXT 37 BYTES INTO RD ARRAY
      NBYTE = NMBYTE
      DO 20 I=1,37
      CALL MOVE (INBUF,NBYTE,RD(I),1,1)
      NBYTE=NBYTE+1
 20   CONTINUE
C
C *****
      IF (NMBYTE.GE.534) KFLG=1
      IF(NMBYTE.GE.534) NMBYTE = 1
C
      CALL DECODE (RD,VAR)
C  *****
      RETURN
      END
```

@PRT,S V.DECODE

$A*LIB(1).DECODE

```
      SUBROUTINE DECODE (RD,VAR)
C
C  THIS PROGRAM DECODES THE DATA BROKEN OUT
C  BY SUBROUTINE DATSTR.  THE DECODED VARIABLES
C  ARE STORED IN ARRAY VAR.
C
      REAL RD(1),VAR(1)
      INTEGER I,T(17)
C
C  CONSTRUCT SCALAR WORDS
C
      DO 5 I=2,8,2
      T(I)=FLD(0,8,RD(I-1))
 5    FLD(20,8,T(I))=FLD(0,8,RD(I))
C
C  COMPUTE SCALAR QUANTITIES
C
C  PRESSURE
C
C*****************************************************************
C*****************************************************************
C***                                                            **
C***   PRESSURE CALIBRATION OF 10/8/81 , 10/9/81 APPLIED        **
C***                                                            **
C*****************************************************************
C*****************************************************************
C
      VAR(1)=T(2)/200.
      VAR(1)= 0.9989789*VAR(1)+0.20325
      IF (FLD(7,1,RD(9)).EQ.1) VAR(1)=-VAR(1)
C
C  SLOW TEMP.
      VAR(2)=T(4)/2000.
      IF (FLD(6,1,RD(9)).EQ.1) VAR(2)=-VAR(2)
```

```
C
C  CONDUCTIVITY
      VAR(3)=T(6)/1000.
C
C  FAST TEMP.
      VAR(4)=T(8)/2000.
      IF (FLD(5,1,RD(9)).EQ.1) VAR(4)=-VAR(4)
C
C   CONSTRUCT VECTOR COMPONENT WORDS
C
      DO 10 I=10,26,2
      T(I-9)=FLD(0,6,RD(I))
      FLD(22,6,T(I-9))=FLD(0,8,RD(I+1))
10    IF (FLD(7,1,RD(I)).EQ.1) T(I-9)=-T(I-9)
C
C  COMPUTE VECTOR COMPONENTS
C
C   VELOCITY VECTOR
C
C************************************************************
C************************************************************
C***                                                      ***
C***  NOTE : NO CALIBRATIONS HAVE BEEN APPLIED TO THE VELOCITY  ***
C***          COMPONENTS IN INSTRUMENT COORDINATES.       ***
C***                                                      ***
C************************************************************
C************************************************************
C
      VAR(5)=T(1)*.0061039
      VAR(6)=-T(3)*.0061039
      VAR(7)=T(5)*.0061039
C
C   MAGNETIC VECTOR
      VAR(8)=-T(7)/1000.
      VAR(9)=T(9)/1000.
      VAR(10)=T(11)/1000.
C
C   ACCELERATION VECTOR
      VAR(11)=T(13)/1000.
      VAR(12)=-T(15)/1000.
      VAR(13)=T(17)/1000.
C
C  COMPUTE CAST TIME
C
C  JULIAN DAY
      VAR(14)=100*FLD(0,4,RD(37))+10*FLD(4,4,RD(37))
     $      +FLD(0,4,RD(36))
C
C  HOUR
      VAR(15)=10*FLD(4,4,RD(36))+FLD(0,4,RD(35))
C
C  MINUTE
      VAR(16)=10*FLD(4,4,RD(35))+FLD(0,4,RD(34))
C
C  SECOND
      VAR(17)=10.*FLD(4,4,RD(34))+FLD(0,4,RD(33))
     $      +FLD(4,4,RD(33))/10.+FLD(0,4,RD(32))/100.
     $      +FLD(4,4,RD(32))/1000.
C
      RETURN
      END
```

@PRT,S V.MOVE

```
@A*LIB(1).MOVE
              SUBROUTINE MOVE (FROM,IFBYTE,TO,ITBYTE,NBYTES)
      C
      C  THIS ROUTINE MOVES BYTES FROM AN INPUT ARRAY
      C  -FROM-, TO AN OUTPUT ARRAY -TO-.  IFBYTE IS
      C  THE BYTE NUMBER OF THE FIRST BYTE TO BE TRANSFERRED
      C  FROM THE INPUT ARRAY.  ITBYTE IS THE BYTE NUMBER
      C  OF THE POSITION IN THE OUTPUT ARRAY WHICH WILL
      C  RECEIVE THE TRANSFERRED BYTE.  NBYTES IS THE
      C  NUMBER OF BYTES TO BE TRANSFERRED.
      C
              DIMENSION FROM(1),TO(1),MASK(10),IBM(9)
              DATA IBM/28,20,12,4,0,24,16,8,0/
              DATA MASK/268435455,-267386880,-1044480,-4080,-15,
           $  -4278190080,-16711680,-65280,
           $  -255,4294967295/
              DO 1000 NM=1,NBYTES
      C
      C.....GET A BYTE
      C
              NB=(NM-1)+IFBYTE
              IN=(NB-1)*8/36+1
              IF((NB-5)/9*9.EQ.(NB-5)) GO TO 100
              IBS=(NB-1)*8-(IN-1)*36
              IF (IBS.EQ.0) IBS=0
              IND=FLD(IBS,8,FROM(IN))
              GO TO 200
      100     IND=FLD(32,4,FROM(IN))*16+FLD(0,4,FROM(IN+1))
      200     CONTINUE
      C
      C.....PUT A BYTE
      C
              NB=(NM-1)+ITBYTE
              IBS=MOD(NB-1,9)+1
              NEL=MOD(NB-1,9)+1
              IN=(NB-1)*8/36+1
              IF ((NB-5)/9*9.EQ.(NB-5)) GO TO 300
              IND=IND*2**IBM(IBS)
              TO(IN)=AND(TO(IN),MASK(NEL))
              TO(IN)=OR(TO(IN),IND)
              GO TO 400
      300     TO(IN)=AND(TO(IN),MASK(5))
              TO(IN)=OR(TO(IN),FLD(28,4,IND))
              TO(IN+1)=AND(TO(IN+1),MASK(10))
              TO (IN+1)=OR(TO(IN+1),FLD(32,4,IND)*2**32)
      400     CONTINUE
      C
      1000    CONTINUE
              RETURN
              END

@PRT,S V.MAPIRA

@A*LIB(1).CONVERSION
      C  MAIN ROUTINE FOR CONVERSION PROGRAM.
      C
      C  THIS PROGRAM CONVERTS THE VELOCITY,
      C  ACCELERATION,AND MAGNETIC VECTORS
      C  FROM INSTRUMENT TO GEOMAGNETIC COORDINATES.
      C
      C  ****************************************************
      C
```

```
C   COMMONS FOR ZREAD
          COMMON/RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(15)
          COMMON/RDATA/VR(15,500)
          COMMON/RDOCF/FDOCR(40)
          COMMON/RDOCI/IDOCR(20)
          COMMON/RDOCA/ADOCR(50)
C
          COMMON/DIAGS/MSGR,MSGU,NNNR,NNNU,NNIP,NNF,NNI,NNA,IRST,IUST
C
C   INITIALIZE CONTROL HEADER
          DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
          DATA NNNR,NNIP,NNF,NNI,NNA/500,15,40,20,50/
          MSGR=0
          MSGU=0
C
C   *****
C
C   COMMONS FOR CONVERSION
          COMMON/BETA/BHAT(3,3),ORTHOG(4,3),ACCAL(3)
          COMMON NCYCLE,ACC(3),XMAG(3),U(3),A(3),H(3)
C
C   INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
          DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
     $    -.055,-.803,1.58,-.794/
          DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
     $    -.197,-.684,-.011,.759/
          DATA  ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
     $    .612,.849,-.071,.609/
C
C   INITIALIZE ACCELEROMETER CALIBRATION COEFFICIENTS
          DATA ACCAL(1),ACCAL(2),ACCAL(3)/-1.000,1.000,1.000/
C
C   ****************************************************
C
   10     CONTINUE
          CALL ZREAD (10,IF,0)
C
C   CHANGE VARIABLE NAMES
          IPR(5)='VLOCG1'
          IPR(6)='VLOCG2'
          IPR(7)='VLOCG3'
          IPR(8)='MAGG1'
          IPR(9)='MAGG2'
          IPR(10)='MAGG3'
          IPR(11)='ACCLG1'
          IPR(12)='ACCLG2'
          IPR(13)='ACCLG3'
C
C   ADD TO ADOC
          ADOCR(37)=' GEOMA'
          ADOCR(38)='G COOR'
          ADOCR(39)='D     '
C
          NCYCLE=0
C
   20     CONTINUE
          NCYCLE=NCYCLE+1
          IF (NCYCLE.GT.NR) GO TO 100
C
C
   50     CONTINUE
C
C   CALL VECTOR CONVERSION SUBROUTINES
```

```
      CALL VCTCON
C
C *****
C WRITE NEW SEGMENT
C
100   CONTINUE
      IF (NCYCLE.LE.NR) GO TO 20
      CALL ZWRIT (-20,IF,G)
C
C CHECK FOR END OF CAST
      IF (IDOCR(1).NE.1) GO TO 10
C
C ***************************************************
      END
```

```
@PRT,S V.CONVVECTOR
```

```
3A*LIB(1).CONVVECTOR
      SUBROUTINE VCTCON
C
C THIS SUBROUTINE CONVERTS THE VELOCITY, ACCELERATION,
C AND MAGNETIC VECTORS FROM INSTRUMENT COORDINATES
C INTO GEOMAGNETIC COORDINATES.
C
C ***********************************************
C COMMONS FOR CONVERSION
      COMMON/BETA/BMAT(3,3),ORTHOG(4,3),ACCAL(3)
      COMMON/RDATA/VR(15,500)
      COMMON/RDOCF/FDOCR(40)
      COMMON/RDOCI/IDOCR(20)
C
      COMMON NCYCLE,ACC(3),XMAG(3),U(3),A(3),H(3)
     REAL V(3),LENGTH
C
C *************************************************
C
C
C *****
C
C CALL SUBROUTINE TO CREATE TRANSFORM MATRIX
      CALL TRNMTX
C
C *****
C
C DO COORDINATE CONVERSIONS
      DO 30 I=1,3
      U(I)=0
      A(I)=0
      H(I)=0
      DO 30 J=1,3
      U(I)=U(I)+BMAT(I,J)*VR(J+4,NCYCLE)
      A(I)=A(I)+BMAT(I,J)*VR(J+10,NCYCLE)
30    H(I)=H(I)+BMAT(I,J)*VR(J+7,NCYCLE)
         WRITE(6,2001) U(1),U(2),U(3)
2001     FORMAT(' V - AFTER :',3G20.7)
            LENGTH=SQRT(U(1)*U(1)+U(2)*U(2)+U(3)*U(3))
            WRITE(6,1001) LENGTH
1001        FORMAT(' ABSOLUTE VELOCITY AFTER TRANSFORM =',G20.5//)
C
C PUT NEW VALUES INTO DATA ARRAY
      DO 50 I=1,3
      VR(I+4,NCYCLE)=U(I)
```

```
            VR(I+10,NCYCLE)=A(I)
      50    VR(I+7,NCYCLE)=H(I)
      C
      C **************************************************
            RETURN
            END

@PRT,S V.TRANSFORMS

$A*LIB(1).TRANSFORMS
             SUBROUTINE TRNMTX
      C
      C  THIS SUBROUTINE CALIBRATES THE ACCELERATION VECTOR,
      C  NORMALIZES THE ACCELERATION AND MAGNETIC VECTORS,
      C  AND CREATES THE TRANSFORM MATRIX 'BMAT'.
      C
      C  **************************************************
      C
      C  COMMONS FOR CONVERSION
             COMMON/RDATA/VR(15,500)
             COMMON/RDOCF/FDOCR(40)
             COMMON/BETA/BMAT(3,3),ORTHOG(4,3),ACCAL(3)
      C
             COMMON NCYCLE,ACC(3),XMAG(3),U(3),A(3),H(3)
      C
      C  **************************************************
      C
      C  CALIBRATE ACCELERATION VECTOR
             DO 10 I=1,3
      10     VR(I+10,NCYCLE)=ACCAL(I)*VR(I+10,NCYCLE)
      C
      C  *****
      C
      C  NORMALIZE 'A' AND 'H' VECTORS
             XNA=0
             XNH=0
             DO 20 I=1,3
             XNA=XNA+VR(I+10,NCYCLE)**2
      20     XNH=XNH+VR(I+7,NCYCLE)**2
      C
             XNA=SQRT(XNA)
             XNH=SQRT(XNH)
      C
             DO 30 I=1,3
             ACC(I)=VR(I+10,NCYCLE)/XNA
      30     XMAG(I)=VR(I+7,NCYCLE)/XNH
      C
      C  CALCULATE DIP FROM 'A' AND 'H'
             SINDIP=0
             DO 40 I=1,3
      40     SINDIP=SINDIP+ACC(I)*XMAG(I)
             DIP=ASIN(SINDIP)
             COSDIP=COS(DIP)
      C
      C  *****
      C
      C  CREATE TRANSFORM MATRIX
             DO 50 I=1,3
      50     BMAT(3,I)=-ACC(I)
      C
             DO 70 I=1,3
      70     BMAT(2,I)=(XMAG(I)+SINDIP*BMAT(3,I))/COSDIP
```

```
      C
            XT2=1.D-BMAT(2,1)**2-BMAT(3,1)**2
            IF (XT2.LT.0) XT2=-XT2
            BMAT(1,1)=SQRT(XT2)
            IF (XMAG(2).LT.0) BMAT(1,1)=-BMAT(1,1)
      C
            XT=ACC(2)*XMAG(3)-ACC(3)*XMAG(2)
            BMAT(1,2)=-BMAT(1,1)*(ACC(1)*XMAG(3)-ACC(3)*XMAG(1))/XT
            BMAT(1,3)=BMAT(1,1)*(ACC(1)*XMAG(2)-ACC(2)*XMAG(1))/XT
      C
      C*************************************************************
      C
            RETURN
            END
```

```
@PRT,S V.MAPCON
```

```
)A*LIB(1).PREFIX
      C***************************************************************
      C PROGRAM               V.PREFIX                                *
      C PURPOSE : TO 1. COMPUTE WI FROM PRESSURE DERIVATIVES   AND     *
      C              2. ATTEMPT TO FIX EFFECTS OF OVERRA  ING          *
      C                                                               *
      C***************************************************************
      C
      C
      C
      C
              COMMON /KBUFF/ KSEG,KNR,KREAL
            COMMON/BETA/UNORTH(4,3),ORTHOG(4,3),W(3),Q(3)
      C  COMMCNS FOR ZREAD
            COMMON/RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(15)
            COMMON/RDATA/VR(15,500)
            COMMON/RDOCF/FDOCR(40)
            COMMON/RDOCI/IDOCR(20)
            COMMON/RDOCA/ADOCR(50)
      C
            COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      C
      C  COMMONS FOR ZWRIT
      C
            COMMON/WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(15)
            COMMON/WDATA/VW(15,500)
            COMMON/WDOCF/FDOCW(40)
            COMMON/WDOCI/IDOCW(20)
            COMMON/WDOCA/ADOCW(50)
      C
      C
              REAL VROT(5000)
      C
      C**************************************************************
      C  INITIALIZE COMMONS FOR ZREAD AND ZWRITE                     *
      C**************************************************************
      C
            DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
            DATA LW,NW,NFW,NIW,NAW/15,500,40,20,50/
            DATA NNNR,NNNW,NNIP,NNF,NNI,NNA/500,500,15,40,20,50/
      C
      C**************************************************************
      C INITIALIZE MATRIX FOR CONVERSION BACK TO UNORTHOGONAL ACOUSTIC*
      C AXES COORDINATES.                                            *
      C**************************************************************
```

```
      C
            CATA UNORTH(1,1),UNORTH(2,1),UNORTH(3,1),UNORTH(4,1)/
           $ .534,.02485,-.67493,.65760/
            DATA UNORTH(1,2),UNORTH(2,2),UNORTH(3,2),UNORTH(4,2)/
           $ .3505,.66162,.01352,.63666/
            CATA  UNORTH(1,3),UNORTH(2,3),UNORTH(3,3),UNORTH(4,3)/
           $ .2268,.03198,.70948,.60185/
      C
      C*************************************************************
      C INITIALIZE THE ORTHOGONALIZATION MATRIX                     *
      C*************************************************************
      C
            DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
           $ -.055,-.803,1.58,-.794/
            DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
           $ -.197,-.684,-.011,.759/
            DATA  ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
           $ .612,.849,-.371,.809/
            MSGR=1
            MSGW=1
      C*************************************************************
      C*   ENTER INPUT AND OUTPUT FEB FILE INFORMATION               *
      C*************************************************************
      C
            WRITE(6,200)
        200 FORMAT(/' ENTER: NUIN1,NSEG1,NSSEG1')
            READ(5,300) NUIN1,NSEG1,NSSEG1
        300 FORMAT()
            WRITE(6,400) NUIN1,NSEG1,NSSEG1
        400 FORMAT(' NUIN1 = ',I2,' NSEG1 = ',I5,' NSSEG1 = ',I5)
        700 FORMAT(' NUIN1 = ',I2,' NSEG2 = ',I5,' NSSEG2 = ',I5,
           *        ' MSGLI2 = ',I2)
      C
      C      READ SPECIFICATIONS FOR OUTPUT FILE
      C
            WRITE(6,900)
        900 FORMAT(' ENTER: NUOUT')
            READ(5,300) NUOUT
            WRITE(6,9999) NUOUT
       9999 FORMAT(' NUOUT = ',I2)
      C
      C*************************************************************
      C*    BEGIN MAIN SECTION OF PROGRAM HERE                      *
      C*                                                            *
      C*         WI WILL REPLACE THE INITIAL TIME (VAR #1*)         *
      C*************************************************************
      C
               NN = 8                         @ HALF WIDTH OF DERIVATIVE SMOO
               NX = 2*NN+1                    @ SET SIZE OF DATA WINDOW
               DELT= 0.0625                   @ NOMINAL TIME DIFFERENCE
               K = NN
               K1 = K*(K+1)*(2*K+1)
               FACTOR=99.55*3.0/(K1*DELT)     @ NORMALIZATION FACTOR FOR
      C                                       @ DP/DT & WI CALCULATION.
               CALL ZBUFF($1,$3,NUIN1,NX,NSSEG1,VROT)
      3        IF(IDOCR(1).EQ.1)IDOCW(1)=1    @ SET END OF CAST FLAG IF REQ.
      C
      C***************************************************************
      C SET THE OUTPUT DOC. BLOCKS                                   *
      C***************************************************************
      C
              DO 7000 IP=1 , 40
                 FDOCW(IP)=FDOCR(IP)
```

```
 7000     CONTINUE
          DO 7001 IP=1,20
 7001     IDOCW(IP)=IDOCF(IP)
          DO 7002 IP= 1,50
 7002     ADOCW(IP)=ADOCF(IP)
          DO 7003 IP=1,15
 7003     IPW(IP)=IPR(IP)
          IPW(14) = 'WI     '
          NMBW = NMBR
          NMFW = NMFR
C
            NPOINT = NSEG1*500         &MAXIMUM POSS. NO. OF DATA CYCLES
C
C
C*****************************************************************
C*****************************************************************
C***                                                           ***
C***                BEGIN THE MAIN LOOPS                       ***
C***                                                           ***
C*****************************************************************
C*****************************************************************
C
C
            DO 1000 I =  1 , NSEG1
             DO 3000 II= 1,500
              DO 3000 JJ = 1,15
               VW(JJ,II) = VR(JJ,II)
 3000       CONTINUE
               DO 1001 J = 1, 500
                   ICYC = (I-1)*500+J
               IF( ICYC .LE. NN+1) IPTR= ICYC
               IF( ICYC .GT. NN+1) IPTR= NN+1
C
C******************************************************************
C         COMPUTE WI AT THE I-TH POINT ( APPROXIMATELY) PLACE IN   *
C         VARIABLE 14 IN THE OUTPUT BUFFER                         *
C******************************************************************
C
                DPDT = 0
               DO 2000 K = 1,NN
                  LP=NN+1+K
                  LM =NN+1-K
                  DPDT=DPDT+K*(VRX(1,LP)-VRX(1,LM))
 2000          CONTINUE
               DPDT=FACTOR*DPDT
               VW(14,J) = DPDT
C
C*****************************************************************
C         TEST FOR GROSS INEQUALITY OF WI AND W. ATTEMPT TO       *
C         CORRECT W BY                                            *
C                  1. BACK TRANSFORMING U,V,W TO Q1,Q2,Q3         *
C                  2. REPLACING Q1 AND Q3 BY .657WI-.534 AND      *
C                     .60185WI-.227, RESPECTIVELY AND             *
C                  3. RETRANSFORMING THE Q COMPONENTS TO U,V,AND W*
C******************************************************************
C
          IF( ABS(DPDT-VRX(7,IPTR)).LT.42)
     1    GOTO 5000                             & TEST FOR BAD VERT.VEL.
C*************************************************************
C* CONVERT VELOCITIES BACK TO ORIGINAL ACOUSTIC COORDS      *
C*************************************************************
C
               DO 2500 K=1,3
```

```
                          W(K) = VRX(K+4,IPTR)
      2500            CONTINUE
                      DO 2600 K = 1,3
                        V = -UNORTH(1,K)
                          DO 2601 JJ= 1,3
                            V = V+W(JJ)*UNORTH(JJ+1,K)
      2601                CONTINUE
                        Q(K) = V
      2600            CONTINUE
      C
                    Q(1) = 0.658*DPDT+ 0.534 @ CORRECT 1-ST ACOUSTIC AXIS
                    Q(3) = 0.602*DPDT+ 0.227 @ CORRECT 3-RD ACOUSTIC AXIS
      C
      C*************************************************************
      C   RETRANSFORM BACK TO ORTHOGONAL COORDINATES                *
      C*************************************************************
      C
                    DO 2610  KK= 1,3
                        W(KK) = ORTHOG(1,KK)
                        DO 2611 LL= 1,3
                          W(KK) = W(KK) + Q(LL)*ORTHOG(LL+1,KK)
      2611              CONTINUE
                    VW(KK+4,J) = W(KK)
      2610          CONTINUE
      5000          CONTINUE                         @EXIT IF W IS GOOD
                    IF(ICYC .GT. NN .OR. I .GT. 1) CALL STEPX($1,$2,1)
      2               CONTINUE                       @IDOC(1)=1 ON READ UNIT
      1001          CONTINUE                         @END OF SEGMENT LOOP
                    IF(I.EQ.NSEG1)IDOCW(1)=1 @ SET END CAST FLAG ON LAST SEG
                    CALL ZWRIT(NUOUT,IF,0) @ WRITE SEGMENT TO OUTPUT FILE
      1000        CONTINUE                           @ END OF MAIN PROCESSING LOOP
                  STOP
      1           CONTINUE                           @EOF RETURN ON READ UNIT
      C
                 IDOCW(1) = 1                        @ SET END OF CAST FLAG
                 CALL ZWRIT(NUOUT,IF,0)              @ WRITE LAST BLOCK
                 STOP
                 END
```

@PRT,S V.ZBUFF

```
ODE331*FCHFILE1(1).VFIX1
      C      THIS PROGRAM COMPUTES THE PROFILER CORRECTED VERTICAL
      C         VELOCITY USING INTEGRATED ACCELERATIONS
      C
      C      BEFORE EXECUTING, THE INPUT AND OUTPUT FEB FILES
      C         MUST BE ASSIGNED TO SOME UNIT NUMBERS
      C
      C      MAPPING ELEMENT:  FCHFILE1.MVFIX1
      C
             DIMENSION A(4,5),X(4),WI(500),AIJ1(4,5),DW(500),DWTIME(500)
             COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
             COMMON /RDATA/VR(10000)
             COMMON /RDOCF/FDOCR(40)
             COMMON /RDOCI/IDOCR(20)
             COMMON /RDOCA/ADOCR(200)
             COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
             COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
             COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NUMA
             COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
             DATA LR,NR,NFR,NIR,NAR/20,500,40,20,50/
             DATA NNNR,NNNW,NNIP,NNF,NNI,NNA/500,500,20,40,20,200/
```

```
C
      GRAV=0.
      ACORR=9.99
      RHOG1=.9945
C
C     GAP= MAXIMUM TIME (SECONDS) BEFORE SYSTEM IS SOLVED
C          AND CORRECTED VALUES ARE COMPUTED
C     GAP1= MAXIMUM TIME GAP (SECONDS) BEFORE LINEAR
C          INTERPOLATION FOR MISSING DATA IS PERFORMED
C
      GAP=1.
      GAP1=0.1
C
      IFIRST=0
C
      WRITE(6,1)
    1 FORMAT(' INPUT UNIT NUMBER, NUMBER OF SEGMENTS, STARTING SEGMENT,
     #MESSAGE LEVEL',
      READ(5,2)IUNR,NSEG,NSSEG,MSGR
    2 FORMAT()
      WRITE (6,3)
    3 FORMAT(' OUTPUT UNIT NUMBER, MESSAGE LEVEL')
      READ(5,2)IUNW,MSGW
C
      NINDEX=0
      IBGN=NSSEG
      DO 100 ISEG=1,NSEG
    4    CALL ZREAD(IUNR,IF,NSSEG)
         IF(IF.NE.0)GO TO 99
         IFLAG=0
         NNSSEG=NSSEG
C
C     LOOP DETERMINES VARIABLE LOCATIONS
C
         DO 5 J=1,LR
            IF(IPR(J).EQ.'VLOCI3')NUMV=J
            IF(IPR(J).EQ.'ACCLI3')NUMA=J
            IF(IPR(J).EQ.'PRESS')NUMP=J
            IF(IPR(J).EQ.'RELSEC')NUMT=J
    5       CONTINUE
C
         INDEX=NINDEX
    8    ISTART=INDEX+1
C
C     CALL SUBROUTINE TO INITIALIZE STARTING VALUES FOR THE
C        FIRST READ OR FOR THE FIRST READ AFTER A TIME GAP
C
         IF(IFIRST.EQ.0)CALL INIT
C
         DO 10 I=ISTART,NR
            IND=(I-1)*LR
            TIMDIF=VR(IND+NUMT)-TLAST
C
C     IF TIME DIFFERENCE TOO LARGE, SOLVE THE SYSTEM
C
            IF(TIMDIF.GT.GAP)GO TO 11
C
            INDEX=INDEX+1
            ACC=VR(IND+NUMA)+ACORR
C
C     IF DATA VALUES ARE MISSING, PERFORM INTERPOLATION
C
            IF(TIMDIF.GE.GAP1)CALL CORREC(IND,WLAST,GAP)
```

```
      C
      C      IF NO DATA VALUES MISSING, PERFORM ACCELERATION
      C         INTEGRATION (F1)
      C
                   IF(TIMDIF.LT.GAP1)F1=F1+(VR(IND+NUMA)+ACORR+ALAST)*TIMDIF/2.
      C
      C      SAVE LAST VALUES FOR NEXT CYCLE
      C
                   TLAST=VR(IND+NUMT)
                   ALAST=VR(IND+NUMA)+ACORR
                   WLAST=VR(IND+NUMV)
                   PLAST=VR(IND+NUMP)
      C
      C      T2= TOTAL TIME
      C      W= MEASURED VELOCITY (CONVERTED TO METERS/SEC)
      C      WBAR= AVERAGE VELOCITY (BASED ON PRESSURE CHANGE)
      C
                   F2=TLAST-T1
                   W=GRAV*F2+VR(IND+NUMV)/100
                   WBAR=0.
                   IF(F2.NE.0.)WBAR=-(PRESS1-VR(IND+NUMP))*RHOG1/F2
      C
      C      SUM NEW VALUES TO ARRAYS IF NOT ALREADY DONE THROUGH
      C         THE CORREC SUBROUTINE
      C
                   IF(TIMDIF.LT.GAP1)CALL ARRAYS
      C
       10          CONTINUE
                GO TO 90
      C
      C      PROGRAM CONTINUES HERE IF A TIME GAP IS DISCOVERED
      C
       11       NINDEX=INDEX
      C
      C      TAKE CARE OF THE CASE WHERE A TIME GAP OCCURS BETWEEN SEGMENTS
      C
                IF(INDEX.EQ.0)NNSSEG=NNSSEG-1
                IF(INDEX.EQ.0)NINDEX=NR
      C
      C      CALL SUBROUTINES TO SOLVE THE SYSTEM AND COMPUTE CORRECTIONS
      C
                CALL SOLVE(PLAST)
                CALL WICOMP(ACORR,NINDEX,NNSSEG,IBGN,IFLAG,GAP1)
      C
       90       NSSEG=NSSEG+1
      C
      C      IF SYSTEM HAS NOT BEEN SOLVED, RESET INDEX AND SET FLAG
      C         FOR VARIABLE INITIATION
      C
                IF(IFLAG.EQ.0)NINDEX=0
                IF(IFLAG.EQ.0)IFIRST=1
      C
      C      IF A COMPLETE SEGMENT HAS NOT BEEN READ, RETURN TO READ
      C        WITHOUT INCREMENTING THE LOOP COUNTER
      C
                IF(INDEX.LT.NR)GO TO 4
      C
       100      CONTINUE
      C
      C      SOLVE SYSTEM AND COMPUTE CORRECTIONS AFTER LAST READ
      C
             NINDEX=INDEX
             NNSSEG=NSSEG-1
```

```
              CALL SOLVE(PLAST)
              CALL WICOMP(ACORR,NINDEX,NNSSEG,IBGN,IFLAG,GAP1)
              GO TO 999
   99         WRITE(6,7)
   7          FORMAT(/,' READ ERROR')
   999        END

@DF*31*FCHFILE1(1).INIT/VFIX1
      C       THIS SUBROUTINE INITIATES DATA FOR THE BEGINNING OF THE
      C          PROGRAM OR AFTER A TIME GAP HAS OCCURRED
      C
              SUBROUTINE INIT
              DIMENSION A(4,5),WI(500),AIJ1(4,5),X(4),DW(500),DWTIME(500)
              COMMON /RDATA/VR(10000)
              COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
              COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
              COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NUMA
              COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,N
      C
              INDEXS=INDEX+1
              IDW=0
              IND=(ISTART-1)*LR
      C
      C       T1,PRESS1= START TIME AND PRESSURE FOR INITIAL READ OR AFTER GAP
      C
              T1=VR(IND+NUMT)
              TLAST=T1
              PRESS1=VR(IND+NUMP)
              F1=0.
              WCHG=0.
              ALAST=0.
              WSAVE=0.
              TSAVE=T1
              ASAVE=0.
      C
      C       LOOP ZEROES ARRAY VALUES
      C
              DO 10 I=1,4
                 DO 10 J=1,5
                    AIJ1(I,J)=0.
   10               A(I,J)=0.
              RETURN
              END
@DF*31*FCHFILE1(1).CORREC/VFIX1
      C       THIS SUBROUTINE INPERPOLATES AND FILLS GAPS FOR MISSING DATA
      C
              SUBROUTINE CORREC(IND,WLAST,GAP)
              COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
              COMMON /RDATA/VR(10000)
              COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
              COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NUMA
              COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,N
              DIMENSION A(4,5),WI(500),AIJ1(4,5),X(4),DW(500),DWTIME(500)
      C
              DW1=VR(IND+NUMV)-WLAST
              SLOPE=DW1/TIMDIF
      C
      C     COMPUTE THE MAXIMUM NUMBER OF TIMES TO LOOP IN ORDER TO DETERMINE THE
      C          NUMBER OF DATA POINTS MISSING (BASED ON AN OPTIMUM
      C           TIME DIFFERENCE OF 0.0625 SECONDS)
      C
              LOOP=INT(GAP/0.0625)+1
```

```
      CHKPTS=0
C
C     LOOP DETERMINES THE NUMBER OF POINTS MISSING
C
      DO 10 I=1,LOOP
         PTS=ABS(TIMDIF/I-0.0625)
         IF(PTS.LE.CHKPTS)NPTS=I
   10    CHKPTS=PTS
C
C     DIVIDE THE TIME INCREMENT EQUALLY AND COMPUTE THE
C        VELOCITY INCREASE PER INCREMENT
C
      TIMDIF=TIMDIF/NPTS
      DW2=SLOPE*TIMDIF
C
C     LOOP FINDS INTERPOLATED VALUES OF F1, W, F2, AND SUMS TO ARRAYS
C
      DO 20 I=1,NPTS
         F1=F1+DW2/100
         W=(WLAST+DW2*I)/100
         F2=F2+TIMDIF
         CALL ARRAYS
   20    CONTINUE
C
C     RESET TIME DIFFERENCE
C
      TIMDIF=TIMDIF*NPTS
      RETURN
      END

DDF31*FCHFILE1(1).ARRAYS/VFIXI
C     THIS SUBROUTINE COMPUTES VALUES IN THE MATRIX USED TO
C        SOLVE FOR ALPHA, BETA, GAMMA, AND LAMBDA
C
      SUBROUTINE ARRAYS
      DIMENSION A(4,5),WI(500),AIJ1(4,5),AIJ2(4,5),DW(500),DWTIME(500)
      DIMENSION X(4)
      COMMON /DATA/A,TI,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESSI,WI,F1,X
      COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NUMA
      COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
C
C     ARRAY AIJ2 STORES VALUES FOR THIS DATA POINT
C     ARRAY AIJ1 STORES VALUES FOR THE LAST DATA POINT
C
      AIJ2(1,1)=F1*F1
      AIJ2(1,2)=F1
      AIJ2(1,3)=-F1*F2
      AIJ2(1,4)=F1/2
      AIJ2(1,5)=F1*W
      AIJ2(2,1)=F1
      AIJ2(2,2)=1.
      AIJ2(2,3)=-F2
      AIJ2(2,4)=0.5
      AIJ2(2,5)=W
      AIJ2(3,1)=F1*F2
      AIJ2(3,2)=F2
      AIJ2(3,3)=-F2*F2
      AIJ2(3,4)=F2/2
      AIJ2(3,5)=F2*W
      AIJ2(4,1)=F1
      AIJ2(4,2)=1.
      AIJ2(4,3)=-F2
      AIJ2(4,4)=0.
```

```
              AIJ2(4,5)=WBAR*GRAV*F2
      C
      C       LOOP INTEGRATES AIJ2 VALUES INTO ARRAY A, AND STORES
      C          AIJ2 VALUES INTO ARRAY AIJ1
      C
              DO 10 I=1,4
                 DO 10 J=1,5
                    A(I,J)=A(I,J)+(AIJ1(I,J)+AIJ2(I,J))*TIMDIF/2
                    AIJ1(I,J)=AIJ2(I,J)
         10         CONTINUE
              A(4,5)=WBAR*F2
              RETURN
              END

FRT,S FCHFILE1.SOLVE/VFIX1,.WICOMP/VFIX1,.WRITE/VFIX1

ODE331*FCHFILE1(1).SOLVE/VFIX1
      C       THIS SUBROUTINE SOLVES THE MATRIX EQUATION AND OUTPUTS
      C          START TIMES AND END TIMES FOR TIME AND PRESSURE FOR
      C          THE PERIOD SINCE THE LAST TIME GAP.  ALSO OUTPUT ARE
      C          ALPHA, BETA, GAMMA, AND LAMBDA
      C
              SUBROUTINE SOLVE(PLAST)
              DIMENSION AA(4,5),A(4,5),X(4),WI(500),AIJ1(4,5)
              COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
              COMMON /RDATA/VR(10000)
              COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
              COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NJMA
      C
              WRITE(6,51)T1,TLAST,PRESS1,PLAST
         51   FORMAT(' START TIME, END TIME         ',2F20.4,/,
             # ' START PRESSURE, END PRESSURE',2F20.4,/)
              DO 20 I=1,4
                 DO 20 J=1,5
         20         AA(I,J)=A(I,J)
              XMAT=SIMUL(4,AA,X,0,0,4)
              WRITE(6,14)XMAT
         14   FORMAT(' XMAT =',G10.5)
              WRITE(6,6)X(1),X(2),X(3),X(4)
          6   FORMAT(' ALPHA =',G10.5,3X,'BETA =',G10.5,3X,'GAMMA =',G10.5,
             #3X,'LAMBDA =',G10.5)
              RETURN
              END

ODE331*FCHFILE1(1).WICOMP/VFIX1
      C       THIS SUBROUTINE COMPUTES THE CORRECTIONS TO VELOCITY
      C
              SUBROUTINE WICOMP(ACORR,NINDEX,NNSSEG,IBGN,IFLAG,GAP1)
              DIMENSION A(4,5),X(4),WI(500),AIJ1(4,5),DW(500),DWTIME(500)
              COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
              COMMON /RDATA/VR(10000)
              COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
              COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NJMA
              COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
      C
              ISTART=INDEXS
              IFIRST=0
              IEND=NNSSEG-IBGN+1
              DO 10 IWRITE=1,IEND
      C
      C       READ SEGMENTS BEGINNING WITH START SEGMENT OR THE LAST
      C          SEGMENT WHERE A TIME GAP OCCURRED
```

```
      C
            CALL ZREAD(IUNR,IF,IBGN)
      C
      C     SET LAST CYCLE TO READ
      C
            INDEX1=NR-1
            IF(IWRITE.EQ.IEND)INDEX1=NINDEX-1
      C
            IND=(ISTART-1)*LR
            CHGTIM=VR(IND+NUMT)-TSAVE
      C
      C     COMPUTE INSTRUMENT VELOCITY (WI) FOR THE FIRST CYCLE
      C     (WI VALUES ARE COMPUTED BY EITHER INTEGRATION OF
      C       ACCELERATION OR BY CHANGE IN VELOCITY, DEPENDING
      C       ON THE TIME GAP)
      C
            IF(CHGTIM.GE.GAP1)WI(ISTART)=WSAVE+X(1)*(VR(IND+NUMV)-WLASTS)/
     #      1C0-X(3)*CHGTIM
            IF(CHGTIM.LT.GAP1)WI(ISTART)=WSAVE+(X(1)*(ASAVE+VR(IND+
     #        NUMA)+2*ACORR)-2*(GRAV+X(3)))*CHGTIM/2
      C
      C     FOR FIRST CYCLE, WI=BETA
      C
            IF(IFIRST.EQ.0)WI(ISTART)=X(2)
      C
      C     COMPUTE WI VALUES TO END OF SEGMENT OR TO LAST CYCLE READ
      C
            DO 30 I=ISTART,INDEX1
               IND=(I-1)*LR
               CHGTIM=VR(IND+LR+NUMT)-VR(IND+NUMT)
               IF(CHGTIM.GE.GAP1)WI(I+1)=WI(I)+X(1)*(VR(IND+LR+NUMV)-
     #          VR(IND+NUMV))/1C0-X(3)*CHGTIM
               IF(CHGTIM.LT.GAP1)WI(I+1)=WI(I)+(X(1)*(VR(IND+NUMA)+
     #         VR(IND+LR+NUMA)+2*ACORR)-2*(GRAV+X(3)))*CHGTIM/2
   30          CONTINUE
      C
            INDEX2=INDEX1+1
      C
      C     CALL SUBROUTINE TO WRITE CORRECTED VELOCITY TO OUTPUT FILE
      C
            CALL WRITE(IF,WLASTS,INDEX2)
      C
      C     RESET VALUES FOR READING NEXT SEGMENT
      C
            IBGN=IBGN+1
            IFIRST=1
            ISTART=1
   10       CONTINUE
      C
      C     SET VALUES FOR RETURN TO MAIN PROGRAM
      C
            IF(INDEX.NE.0)NSSEG=NNSSEG-1
            IF(INDEX.NE.0)IBGN=IBGN-1
            IFLAG=1
            IFIRST=0
            IF(INDEX.EQ.0)NINDEX=0
            RETURN
            END

ODF*31*FCHFILE1(1).WRITE/VFIX1
      C     THIS SUBROUTINE WRITES CORRECTED VELOCITY VALUES TO A NEW FEB FILE
      C     (THE OUTPUT FILE CONTAINS 2 VARIABLES MORE THAN THE INPUT FILE)
      C
```

```
      SUBROUTINE WRITE(IF,WLASTS,INDEX2)
      COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
      COMMON /RDATA/VR(10000)
      COMMON /RDOCF/FDOCR(40)
      COMMON /RDOCI/IDOCR(20)
      COMMON /RDOCA/ADOCR(200)
      COMMON /WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(20)
      COMMON /WDATA/VW(10000)
      COMMON /WDOCF/FDOCW(40)/WDOCI/IDOCW(20)/WDOCA/ADOCW(200)
      COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      COMMON /DATA/A,T1,NUMT,NUMP,NUMV,ISTART,TLAST,ALAST,PRESS1,WI,F1,X
      COMMON /DATA1/AIJ1,WSAVE,TSAVE,ASAVE,TIMDIF,INDEX,GRAV,IFIRST,NUMA
      COMMON /DATA2/WCHG,DW,DWTIME,IDW,F2,WBAR,NSSEG,INDEXS,IUNR,IUNW,W
      DIMENSION A(4,5),WI(500),AIJ1(4,5),DW(500),DWTIME(500)
C
      DO 10 I=1,NNF
   10    FDOCW(I)=FDOCR(I)
      DO 20 I=1,NNI
   20    IDOCW(I)=IDOCR(I)
      DO 30 I=1,NNA
   30    ADOCW(I)=ADOCR(I)
      NW=NR
      NBW=NBR
      NMBW=NMBR
      NMFW=NMFR
      NFW=NFR
      NIW=NIR
      NAW=NAR
      LW=LR+2
      NUMV1=NUMV-1
C
C     VARIABLES BEFORE VERTICAL VELOCITY ARE WRITTEN OUT THE SAME
C
      DO 40 J=1,NUMV1
         IPW(J)=IPR(J)
         DO 40 I=1,NR
            IND1=(I-1)*LR
            IND2=(I-1)*LW
   40       VW(IND2+J)=VR(IND1+J)
C
C     VARIABLE W=V-WI IS WRITTEN OUT JUST BEFORE THE VERTICAL VELOCITY
C     VARIABLE WI IS WRITTEN OUT AT THE END OF THE INPUT VARIABLES
C
      DO 50 I=ISTART,INDEX2
         IND1=(I-1)*LR
         IND2=(I-1)*LW
         VW(IND2+LW)=WI(I)*100.
   50    VW(IND2+NUMV)=VR(IND1+NUMV)-WI(I)*100
C
C     VARIABLES FROM VERTICAL VELOCITY THROUGH THE LAST INPUT
C        VARIABLE ARE WRITTEN OUT SHIFTED OVER ONE LOCATION
C
      NUMV1=NUMV+1
      LW1=LW-1
      DO 60 J=NUMV1,LW1
         IPW(J)=IPR(J-1)
         DO 60 I=1,NR
            IND1=(I-1)*LR
            IND2=(I-1)*LW
   60       VW(IND2+J)=VR(IND1+J-1)
C
      IPW(NUMV)='W     '
      IPW(LW)='WI    '
```

```
C
C     SAVE VALUES FOR RETURN TO SUBROUTINE WICOMP
C
      WSAVE=WI(NR)
      IND2=(NR-1)*LR
      TSAVE=VR(IND2+NUMT)
      ASAVE=VR(IND2+NUMA)
      WLASTS=VR(IND2+NUMV)
C
C     CALL ZWRIT ONLY IF THE ENTIRE SEGMENT HAS BEEN COMPLETED
C
      IF(INDEX2.LT.NR)RETURN
      CALL ZWRIT(IUNW,IF,0)
      RETURN
      END
```

PK FCHFILE.SIMUL

ODF331*FCHFILE(1).SIMUL

```
C   FUNCTION SIMUL (N,A,X,EPS,INDIC,NRC)
C                                                                       SI
C      WHEN INDIC IS NEGATIVE, SIMUL COMPUTES THE INVERSE OF THE N BY   SI
C      N MATRIX A IN PLACE.  WHEN INDIC IS ZERO, SIMUL COMPUTES THE     SI
C      N SOLUTIONS X(1)...X(N) CORRESPONDING TO THE SET OF LINEAR       SI
C      EQUATIONS WITH AUGMENTED MATRIX OF COEFFICIENTS IN THE N BY      SI
C      N+1 ARRAY A AND IN ADDITION COMPUTES THE INVERSE OF THE          SI
C      COEFFICIENT MATRIX IN PLACE AS ABOVE.  IF INDIC IS POSITIVE,     SI
C      THE SET OF LINEAR EQUATIONS IS SOLVED BUT THE INVERSE IS NOT     SI
C      COMPUTED IN PLACE. THE GAUSS-JORDAN COMPLETE ELIMINATION METHODSI
C      IS EMPLOYED WITH THE MAXIMUM PIVOT STRATEGY.  ROW AND COLUMN     SI
C      SUBSCRIPTS OF SUCCESSIVE PIVOT ELEMENTS ARE SAVED IN ORDER IN    SI
C      THE IROW AND JCOL ARRAYS RESPECTIVELY.  K IS THE PIVOT COUNTER,SI
C      PIVOT THE ALGEBRAIC VALUE OF THE PIVOT ELEMENT, MAX              SI
C      THE NUMBER OF COLUMNS IN A AND DETER THE DETERMINANT OF THE      SI
C      COEFFICIENT MATRIX.  THE SOLUTIONS ARE COMPUTED IN THE (N+1)TH SI
C      COLUMN OF A AND THEN UNSCRAMBLED AND PUT IN PROPER ORDER IN      SI
C      X(1)...X(N) USING THE PIVOT SUBSCRIPT INFORMATION AVAILABLE      SI
C      IN THE IROW AND JCOL ARRAYS.  THE SIGN OF THE DETERMINANT IS     SI
C      ADJUSTED, IF NECESSARY, BY DETERMINING IF AN ODD OR EVEN NUMBERSI
C      OF PAIRWISE INTERCHANGES IS REQUIRED TO PUT THE ELEMENTS OF THESI
C      JORD ARRAY IN ASCENDING SEQUENCE WHERE JORD(IROW(I)) = JCOL(I).SI
C      IF THE INVERSE IS REQUIRED, IT IS UNSCRAMBLED IN PLACE USING     SI
C      Y(1)...Y(N) AS TEMPORARY STORAGE.  THE VALUE OF THE DETERMINANTSI
C      IS RETURNED AS THE VALUE OF THE FUNCTION.  SHOULD THE POTENTIALSI
C      PIVOT OF LARGEST MAGNITUDE BE SMALLER IN MAGNITUDE THAN EPS,     SI
C      THE MATRIX IS CONSIDERED TO BE SINGULAR AND A TRUE ZERO IS       SI
C      RETURNED AS THE VALUE OF THE FUNCTION.                           SI
C
C   REFERENCES CARNAHAN, LUTHER AND WILKES (1969)
C              APPLIED NUMERICAL METHODS. WILEY, NEW YORK.
C
C                                                                       SI
    FUNCTION SIMUL (N,A,X,EPS,INDIC,NRC)
    DIMENSION IROW(50), JCOL(50), JORD(50), Y(50), A(NRC,NRC), X(N)     SI
C                                                                       SI
    MAX = N                                                             SI
    IF ( INDIC.GE.0 )  MAX = N + 1                                      SI
C                                                                       SI
C   .....IS N LARGER THAN 50 .....                                      SI
    IF ( N.LE.50 )   GO TO 5                                            SI
    WRITE( 6 ,200)                                                      SI
    SIMUL = 0.                                                          SI
```

```
          RETURN                                                        SI
C                                                                       SI
C         ..... BEGIN ELIMINATION PROCEDURE .....                       SI
     5    DETER = 1.                                                    SI
          DO 18   K = 1, N                                              SI
          KM1 = K - 1                                                   SI
C                                                                       SI
C         ..... SEARCH FOR THE PIVOT ELEMENT .....                      SI
          PIVOT = 0.                                                    SI
          DO 11   I = 1, N                                              SI
          DO 11   J = 1, N                                              SI
C         ..... SCAN IROW AND JCOL ARRAYS FOR INVALID PIVOT SUBSCRIPTS .....SI
          IF ( K.EQ.1 )   GO TO 9                                       SI
          DO 8   ISCAN = 1, KM1                                         SI
          DO 8   JSCAN = 1, KM1                                         SI
          IF ( I.EQ.IROW(ISCAN)  )    GO TO 11                          SI
     8    IF ( J.EQ.JCOL(JSCAN)  )    GO TO 11                          SI
     9    IF ( ABS(A(I,J)).LE. ABS(PIVOT)  )    GO TO 11                SI
          PIVOT = A(I,J)                                                SI
          IROW(K) = I                                                   SI
          JCOL(K) = J                                                   SI
    11    CONTINUE                                                      SI
C                                                                       SI
C         ..... INSURE THAT SELECTED PIVOT IS LARGER THAN EPS .....     SI
          IF (  ABS(PIVOT).GT.EPS )    GO TO 13                         SI
          SIMUL = 0.                                                    SI
          RETURN                                                        SI
C                                                                       SI
C         ..... UPDATE THE DETERMINANT VALUE .....                      SI
    13    IROWK = IROW(K)                                               SI
          JCOLK = JCOL(K)                                               SI
          DETER = DETER*PIVOT                                           SI
C                                                                       SI
C         ..... NORMALIZE PIVOT ROW ELEMENTS .....                      SI
          DO 14    J = 1, MAX                                           SI
    14    A(IROWK,J) = A(IROWK,J)/PIVOT                                 SI
C                                                                       SI
C         ..... CARRY OUT ELIMINATION AND DEVELOP INVERSE .....         SI
          A(IROWK,JCOLK) = 1./PIVOT                                     SI
          DO 18    I = 1, N                                             SI
          AIJCK = A(I,JCOLK)                                            SI
          IF ( I.EQ.IROWK )    GO TO 18                                 SI
          A(I,JCOLK) = - AIJCK/PIVOT                                    SI
          DO 17    J = 1, MAX                                           SI
    17    IF ( J.NE.JCOLK )    A(I,J) = A(I,J) - AIJCK*A(IROWK,J)       SI
    18    CONTINUE                                                      SI
C                                                                       SI
C         ..... ORDER SOLUTION VALUES (IF ANY) AND CREATE JORD ARRAY .....  SI
          DO 20    I = 1, N                                             SI
          IROWI = IROW(I)                                               SI
          JCOLI = JCOL(I)                                               SI
          JORD (IROWI) = JCOLI                                          SI
    20    IF ( INDIC.GE.0 )    X(JCOLI) = A(IROWI,MAX)                  SI
C                                                                       SI
C         ..... ADJUST SIGN OF DETERMINANT .....                        SI
          IF(N.EQ.1)   GO TO 24
          INTCH = 0                                                     SI
          NM1 = N - 1                                                   SI
          DO 22   I = 1, NM1                                            SI
          IP1 = I + 1                                                   SI
          DO 22    J = IP1, N                                           SI
          IF ( JORD(J).GE.JORD(I))    GO TO 22                          SI
          JTEMP = JORD(J)                                               SI
```

```
          JORD(J) = JORD(I)                                              SI
          JCRD(I) = JTEMP                                                SI
          INTCH = INTCH + 1                                              SI
   22     CONTINUE                                                       SI
          IF ( INTCH/2*2.NE.INTCH )   DETER = - DETER                    SI
C                                                                        SI
C         ..... IF INDIC IS POSITIVE RETURN WITH RESULTS .....           SI
   24     IF ( INDIC.LE.0 )   GO TO 26                                   SI
          SIMUL = DETER                                                  SI
          RETURN                                                         SI
C                                                                        SI
C         ..... IF INDIC IS NEGATIVE OR ZERO, UNSCRAMBLE THE INVERSE     SI
C               FIRST BY ROWS .....                                      SI
   26     DO 28   J = 1, N                                               SI
          DO 27   I = 1, N                                               SI
          IROWI = IROW(I)                                                SI
          JCOLI = JCOL(I)                                                SI
   27     Y(JCOLI) = A(IROWI,J)                                          SI
          DO 28   I = 1, N                                               SI
   28     A(I,J) = Y(I)                                                  SI
C         ..... THEN BY COLUMNS .....                                    SI
          DC 30   I = 1,N                                                SI
          DO 29   J = 1, N                                               SI
          IROWJ = IROW(J)                                                SI
          JCOLJ = JCOL(J)                                                SI
   29     Y(IROWJ) = A(I,JCOLJ)                                          SI
          DO 30   J = 1, N                                               SI
   30     A(I,J) = Y(J)                                                  SI
C                                                                        SI
C         ..... RETURN FOR INDIC NEGATIVE OR ZERO .....                  SI
          SIMUL = DETER                                                  SI
          RETURN                                                         SI
C                                                                        SI
C         ..... FORMAT FOR OUTPUT STATEMENT .....                        SI
  200     FORMAT( 10HON TOO BIG )                                        SI
C                                                                        SI
          END                                                            SI
```

```
PRT,S FCHFILE1.MVFIX1,.VFIX1-S,.ENDS/VFIX1-S,.PRESS/VFIX1-S

 :37'*FCHFILE1(1).VFIX1-S
C       THIS PROGRAM COMPUTES PROFILER CORRECTED VER.ICAL
C          VELOCITY BY USING THE PRESSURE DERIVATIVE
C
C       BEFORE EXECUTING, THE INPUT DATA FEB FILE MUST BE A''IGNED
C          TO SOME UNIT NUMBER, AND AN OUTPUT FILE ASSIGNED.  TIME
C          INTERPOLATION OF THE INPUT FEB FILE SHOULD HAVE BEEN
C          COMPLETED (HTP*PROG.ZINTERP)
C
C        MAPPING ELEMENT:   FCHFILE1.MVFIX1-S
C
        DIMENSION WI(500),P(17)
        COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUN.,NUMV
        COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(..)
        COMMON /RDATA/VR(20000)
        COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
        COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,N..,.RST,IWST
        DATA NNNR,NNNW,NNIP,NNF,NNI,NNA/5000,5000,20,50,5J,.J0/
C
C       SET INITIAL DATA (K= NUMBER OF POINTS ON EACH SI.. OF
C         THE PRESSURE DATA POINT TO BE USED IN COMPU..N. THE
C         DERIVATIVE; DELT= TIME INTERVAL BETWEEN P..NTS)
```

```
C
      K=8
      K1=K+1
      K2=2*K+1
      DELT=.0625
      RHOG1=.9955
      COEFF=3./(K*(K+1)*(2*K+1)*DELT)
      NSGR=0
      NSGW=0
C
      WRITE(6,2)
 2    FORMAT(' INPUT UNIT, OUTPUT UNIT, NUMBER SEGMENTS, START SEGMENT,'
     #,/,' PRESSURE VARIABLE NUMBER, VERTICAL VELOCITY VARIABLE NUMBER')
      READ(5,1)IUNR,IUNW,NSEG,NSSEG,NVAR,NUMV
 1    FORMAT()
      KFLAG=0
      ISTART=K+1
      CALL ZREAD(IUNR,IF,NSSEG)
      IEND=NR
      DO 30 KSEG=1,NSEG
C
C     FOR FIRST SEGMENT, CALL SUBROUTINE TO SET FIRST K DERIVATIVES TO ZERO.
C     FOR LAST SEGMENT, CALL SUBROUTINE TO SET LAST K DERIVATIVES TO ZERO.
C
         IF(KSEG.EQ.1)CALL ENDS(WI)
         IF(KSEG.NE.1)ISTART=1
         IF(KSEG.EQ.NSEG)CALL ENDS(WI)
C
         DO 10 J=ISTART,IEND
C
C     CALL SUBROUTINE TO READ SPAN OF PRESSURE VALUES INTO ARRAY P
C
            CALL PRESS(P,KFLAG)
            KFLAG=1
            SUM=0.
C
C     COMPUTE PRESSURE DERIVATIVE
C
            DO 20 II=-K,K
 20            SUM=SUM+II*P(K1+II)
            DPOT=COEFF*SUM
            WI(J)=RHOG1*DPOT
C
 10         CONTINUE
C
C     CALL SUBROUTINE TO WRITE NEW FEB FILE
C
         IF (KSEG.EQ.NSEG)GO TO 25
         NSSEG=NSSEG-1
         CALL ZREAD(IUNR,IF,NSSEG)
 25      CALL WRITE(WI)
         WRITE(6,3)NSSEG
 3       FORMAT(' COMPLETED SEGMENT #',I6)
         IF(KSEG.EQ.NSEG)GO TO 30
         NSSEG=NSSEG+1
         CALL ZREAD(IUNR,IF,NSSEG)
 30      CONTINUE
      END

E37**FCHFILE1(1).ENDS/VFIX1-S
      C     THIS SUBROUTINE SETS THE FIRST K PRESSURE DERIVATIVES TO ZERO FOR
      C        THE FIRST SEGMENT, AND THE LAST K EQUAL TO ZERO FOR THE LAST
```

```
C        SEGMENT
C
      SUBROUTINE ENDS(WI)
      DIMENSION WI(1)
      COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW,NUMV
      COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
      COMMON /RDATA/VR(20000)
      COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
      COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
      ISTOP=NR-K+1
      II=ISTOP
      III=NR
C
C     SET LAST CYCLE NUMBER FOR LAST SEGMENT
C
      IF(KSEG.EQ.NSEG)IEND=ISTOP-1
C
      IF(KSEG.EQ.1)II=1
      IF(KSEG.EQ.1)III=K
      DO 10 I=II,III
   10    WI(I)=0.
      RETURN
      END
```

E33 *FCHFILE1(1).PRESS/VFIX1-S

```
C     THIS SUBROUTINE READS A SPAN OF PRESSURE INTO ARRAY P
C
      SUBROUTINE PRESS(P,KFLAG)
      DIMENSION P(1)
      COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW,NUMV
      COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
      COMMON /RDATA/VR(20000)
      COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
      COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
      IF(KFLAG.EQ.1)GO TO 11
C
C     FOR FIRST CALL, READ IN 2*K+1 PRESSURE VALUES
C
      DO 10 M=1,K2
         IND=(M-1)*LR+NVAR
         P(M)=VR(IND)
   10    CONTINUE
      RETURN
C
C     FOR CALLS OTHER THAN THE FIRST, SHIFT PRESSURE VALUES DOWN BY ONE
C
   11 K1=K2-1
      DO 20 M=1,K1
   20    P(M)=P(M+1)
C
      IND=IND+LR
      ICHK=LR*NR
C
C      IF AT END OF SEGMENT, CALL SUBROUTINE TO READ IN NEXT SEGMENT
C       AND READ IN LAST PRESSURE VALUE IN SPAN
C
      IF(IND.GT.ICHK)CALL NEXSEG
      P(K2)=VR(IND)
C
      RETURN
      END
```

```
T,S FCHFILE1.NEXSEG/VFIX1-S,.WRITE/VFIX1-S,.MVFIX1-S

E33'*FCHFILE1(1).NEXSEG/VFIX1-S
      C      THIS SUBROUTINE READS THE NEXT SEGMENT AND RESETS THE INDEX
      C
             SUBROUTINE NEXSEG
             COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW
             COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
             COMMON /RDATA/VR(20000)
             COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
             COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      C
             NSSEG=NSSEG+1
             CALL ZREAD(IUNR,IF,NSSEG)
             IND=NVAR
             RETURN
             END

E33'*FCHFILE1(1).WRITE/VFIX1-S
      C      THIS SUBROUTINE WRITES THE NEW FEB FILE
      C       (THE OUTPUT FILE CONTAINS 2 MORE VARIABLES THAN THE INPUT FILE)
      C
             SUBROUTINE WRITE(WI)
             DIMENSION WI(1)
             COMMON KSEG,IEND,K,K2,IND,NVAR,NSEG,NSSEG,IUNR,IUNW,NUMV
             COMMON /RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(20)
             COMMON /RDATA/VR(20000)
             COMMON /RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(200)
             COMMON /DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
             COMMON /WHDR/LW,NW,NBW,NMBW,NMFW,NFW,NIW,NAW,IPW(20)
             COMMON /WDATA/VW(20000)
             COMMON /WDOCF/FDOCW(50)/WDOCI/IDOCW(50)/WDOCA/ADOCW(200)
      C
             DO 10 I=1,NNF
       10       FDOCW(I)=FDOCR(I)
             DO 20 I=1,NNI
       20       IDOCW(I)=IDOCR(I)
             DO 30 I=1,NNA
       30       ADOCW(I)=ADOCR(I)
             NW=NR
             NBW=NBR
             NMBW=NMBR
             NMFW=NMFR
             NFW=NFR
             NIW=NIR
             NAW=NAR
             LW=LR+2
      C
      C      VARIABLES BEFORE VERTICAL VELOCITY ARE WRITTEN OUT THE SAME
      C
             NUMV1=NUMV-1
             DO 40 J=1,NUMV1
                IPW(J)=IPR(J)
                DO 40 I=1,NR
                   IND1=(I-1)*LR
                   IND2=(I-1)*LW
       40          VW(IND2+J)=VR(IND1+J)
      C
      C      VARIABLE W=V-WI IS WRITTEN OUT JUST BEFORE THE VERTICAL VELOCITY.
      C      VARIABLE WI IS WRITTEN OUT AT THE END OF THE INPUT VARIABLES.
      C
             DO 50 I=1,NR
                IND1=(I-1)*LR
```

```
                 IND2=(I-1)*LW
                 VW(IND2+LW)=WI(I)*100
      50         VW(IND2+NUMV)=VR(IND1+NUMV)-WI(I)*100
C
            NUMV1=NUMV+1
            LW1=LW-1
C
C           VARIABLES FROM VERTICAL VELOCITY THROUGH THE LAST INPUT VARIABLE
C             ARE WRITTEN OUT SHIFTED OVER ONE LOCATION
            DO 60 J=NUMV1,LW1
               IPW(J)=IPR(J-1)
               DO 60 I=1,NR
                  IND1=(I-1)*LR
                  IND2=(I-1)*LW
      60          VW(IND2+J)=VR(IND1+J-1)
C
            IPW(NUMV)='W     '
            IPW(LW)='WI    '
            CALL ZWRIT(IUNW,IF,0)
            RETURN
            END
```

SA*LIB(1).UNORTHOG

```
C
C*************************************************************
C PROGRAM     : UNORTHOG                                      *
C PURPOSE     : TO CONVERT PROFILER FILES WITH ORTHOGONAL     *
C               INSTRUMENT VELOCITIES TO FILES WITH THE       *
C               ORIGINAL VELOCITIES IN ACOUSTIC AXES COORDS.  *
C*************************************************************
C
C
            COMMON/BETA/ORTHOG(4,3),W(3)
C  COMMONS FOR ZREAD
            COMMON/RHDR/LR,NR,NBR,NNBR,NMFR,NFR,NIR,NAR,IPR(15)
            COMMON/RDATA/VR(15,500)
            COMMON/RDOCF/FDOCR(40)
            COMMON/RDOCI/IDOCR(20)
            COMMON/RDOCA/ADOCR(50)
C
            COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
C  INITIALIZE CONTROL HEADER
            DATA LR,NR,NFR,NIR,NAR/15,500,40,20,50/
            DATA NNAR,NNIP,NNF,NNI,NNA/500,15,40,20,50/
C  INITIALIZE VELOCITY ORTHOGONALIZATION MATRIX
            DATA ORTHOG(1,1),ORTHOG(2,1),ORTHOG(3,1),ORTHOG(4,1)/
         $  .534,.02485,-.67493,.65760/
            DATA ORTHOG(1,2),ORTHOG(2,2),ORTHOG(3,2),ORTHOG(4,2)/
         $  .3505,.66162,.01352,.63666/
            DATA  ORTHOG(1,3),ORTHOG(2,3),ORTHOG(3,3),ORTHOG(4,3)/
         $  .2268,.03198,.70948,.60185/
C
            MSGR=1
            MSGW=1
C*************************************************************
C*    ENTER INPUT AND OUTPUT FEB FILE INFORMATION             *
C*************************************************************
C
            WRITE(6,200)
      200   FORMAT(/' ENTER: NUIN1,NSEG1,NSSEG1')
            READ(5,300) NUIN1,NSEG1,NSSEG1
      300   FORMAT()
```

```
      WRITE(6,400) NUIN1,NSEG1,NSSEG1
  400 FORMAT(' NUIN1 = ',I2,' NSEG1 = ',I5,' NSSEG1 = ',I5)
  700 FORMAT(' NUIN1 = ',I2,' NSEG2 = ',I5,' NSSEG2 = ',I5,
     *  ' NSGLI2 = ',I2)
C
C     READ SPECIFICATIONS FOR OUTPUT FILE
C
      WRITE(6,900)
  900 FORMAT(' ENTER: NUOUT')
      READ(5,300) NUOUT
      WRITE(6,9999) NUOUT
 9999 FORMAT(' NUOUT = ',I2)
C
C*****************************************************************
C*   BEGIN MAIN SECTION OF PROGRAM HERE                          *
C*****************************************************************
C
C
           DO 1000 LOOP = NSSEG1, NSEG1
               CALL ZREAD(NUIN1,IF,LOOP)
               DO 2000 I= 1,NR
C
C*************************************************************
C* CONVERT VELOCITIES BACK TO ORIGINAL ACOUSTIC COORDS       *
C*************************************************************
C
                 DO 2500 K=1,3
                   W(K) = VR(K+4,I)
C                 IF(I.LT.3) WRITE(6,7001)K,W(K)
7001             FORMAT(' K = ',I3,' W(K) = ',G10.6)
2500           CONTINUE
               DO 2600 K = 1,3
                 V = -ORTHOG(1,K)
                 DO 2601 J= 1,3
                   V = V+W(J)*ORTHOG(J+1,K)
2601             CONTINUE
                 VR(K+4,I) = V
C              IF(I.LT.3) WRITE(6,7002) I,K,V
7002           FORMAT(' I=',I2,' K=',I2,' V(K)=',G10.6)
2600           CONTINUE
2000       CONTINUE
           CALL ZWRIT(-NUOUT,IF,0)
1000       CONTINUE
           STOP
           END
```

```
@PRT,S V.PREFIX

 @A*LIB(1).TSERPLOT2
C   PROGRAM TSERPLOT2
C
C    JIM VEGA   CSC   SEPT. 1981
C
C   THIS PROGRAM PRODUCES PLOTS OF ANY FEB VARIABLE
C   VS EITHER TIME OR CYCLE NUMBER.
C
C *****************************************************************
C    THIS PROGRAM IS EXECUTED BY THE FOLLOWING:
C
C                  @XQT VEGA*LIB.TSERPLOT2
C
C    FOLLOWED BY TWO DATA CARDS FOR EACH SUBPLOT.
```

```
C     DATA CARD 1 CONTAINS:
C
C     IU,IABSIS,IP,YMAX,YSTP,YMIN,IDEC
C
C     WHERE:
C       IU      - LOGICAL UNIT NUMBER OF INPUT FILE.
C       IABSIS  - POINTER DETERMINING ABSCISSA VALUES
C                 0 - CYCLE NUMBER
C                 1 - RELATIVE TIME (IN SECONDS)
C       IP      - POSITION OF ORDINATE VARIABLE IN FEB
C                 DATA ARRAY VR.
C       YMAX*   - MAXIMUM EXPECTED VALUE OF ORDINATE VARIABLE
C       YSTP*   - ORDINATE LABELING INTERVAL
C       YMIN*   - MINIMUM EXPECTED VALUE OF ORDINATE VARIABLE
C                 (VALUE OF ORDINATE AT ORIGIN)
C       IDEC    - DECIMATION
C
C       * - THESE VARIABLES ARE FLOATING POINT RATHER THAN INTEGER
C
C     *** FOR A PLOT AGAINST CYCLE NUMBER (IABSIS=0) ***
C     DATA CARD 2 CONTAINS:
C
C     NUMSEG,IBEGIN,CYCIN
C
C     WHERE:
C       NUMSEG - NUMBER OF SEGMENTS TO BE PLOTTED
C       IBEGIN - SEGMENT NUMBER OF FIRST SEGMENT TO BE PLOTTED
C       CYCIN  - NUMBER OF CYCLES PER INCH OF PLOT
C
C     NUMSEG AND IBEGIN ARE INTEGERS,CYCIN IS FLOATING POINT
C
C     *** FOR A PLOT AGAINST RELATIVE TIME (IABSIS=1) ***
C     DATA CARD 2 CONTAINS:
C
C     ISTSEC,INDSEC,IPTIME
C
C     WHERE:
C       ISTSEC - PLOT START TIME (IN RELATIVE SECONDS)
C       INDSEC - PLOT END TIME (IN RELATIVE SECONDS)
C       IPTIME - LOCATION OF RELATIVE TIME VARIABLE IN
C                THE FEB DATA ARRAY VR
C
C     ALL THREE PARAMETERS ARE INTEGERS.
C
C     THE FOLLOWING DATA CARD SHOULD FOLLOW THE SECOND
C     DATA CARD OF THE LAST SUBPLOT:
C
C     99,0,0,0.,0.,0.,0
C
C     TO INSURE PROPER TERMINATION OF BOTH THE PROGRAM
C     AND THE PLOT.
C
C     ***** NOTES *****
C
C     THIS PROGRAM IS SET UP TO USE THE 34 INCH PAPER
C     ON THE ZETA PLOTTER, AS A RESULT THE MAXIMUM NUMBER
C     OF SUBPLOTS PER PLOT IS 5.
C
C     THIS PROGRAM PRODUCES AN INTERMEDIATE PLOT FILE ON
C     UNIT 25.  IT IS NECESSARY TO EXECUTE ONE OF THE DISSPLA
C     POST-PROCESSORS TO OBTAIN A PLOT TAPE.
C
C     IT IS HIGHLY RECOMMENDED THAT THE INTERMEDIATE PLOT FILE
```

```
C   ON UNIT 25 BE COPIED TO ANOTHER FILE IMMEDIATELY FOLLOWING
C   EACH EXECUTION TO GUARD AGAINST LOSS.
C
C
C*****************************************************************
      WRITE (6,1)
    1 FORMAT (' FOR INSTRUCTIONS TERMINATE RUN AND ADD',
     $ ' VEGA*LIB.INST/PLOT2')
C
C  COMMONS FOR ZREAD
      COMMON/RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(15)
      COMMON/RDOCF/FDOCR(50)/RDOCI/IDOCR(50)/RDOCA/ADOCR(100)
      COMMON/RDATA/VR(15,500)
      COMMON/DIAGS/MSGP,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
      COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IDEC,YORIG
C
C  INITIALIZE FEB READ
      DATA LR,NR,NFR,NIR,NAR/20,500,50,50,100/
      DATA NNNR,NNIP,NNF,NNI,NNA/500,20,50,50,100/
      YORIG=1.00
C
C  SET UP PLOT PAGE
      CALL COMPRS
      CALL BGNPL (0)
      CALL SCMPLX
      CALL NOERDR
      CALL FLATBD
      CALL PAGE (72.0,34.0)
      CALL TITLE (0,0,0,0,0,0,1.0,1.0)
      CALL GRAF (0.0,1.0,1.0,0.0,1.0,1.0)
      CALL ENDGR (0)
C
   20 CONTINUE
C  READ INPUT PARAMETERS
      READ (5,1000) IU,IABSIS,IP,YMAX,YSTP,YMIN,IDEC
      IF (IU.EQ.99) GO TO 998
C
      IF (IABSIS.EQ.1) GO TO 50
C
      READ (5,1000) NUMSEG,IBEGIN,CYCIN
      XAXIS=NUMSEG/(CYCIN/500.)
      IF (XAXIS.GT.71.) GO TO 900
      GO TO 60
C
   50 CONTINUE
      READ (5,1000) ISTSEC,INDSEC,IPTIME
C
   60 CONTINUE
C
      IF (IABSIS.NE.1) CALL CYCLNM (NUMSEG,IBEGIN,CYCIN,XAXIS)
      IF (IABSIS.EQ.1) CALL RELSEC (ISTSEC,INDSEC,IPTIME)
      YORIG=YORIG+6.75
      GO TO 20
C
  900 WRITE (6,901) XAXIS
      GO TO 20
  901 FORMAT (' XAXIS TOO LARGE  XAXIS=',F7.2,/,
     $ ' MAXIMUM ALLOWABLE SIZE IS 71 INCHES',/,
     $ ' SUBPLOT WILL BE OMITTED')
C
  998 CONTINUE
C
```

```
      WRITE(6,777)
 777  FORMAT(' CALL DONEPL')
      CALL DONEPL
1000  FORMAT ()
C
      END

@PRT,S V.CYCLNM/PLOT2

GA*LIB(1).CYCLNM/PLOT2
      SUBROUTINE CYCLNM (NUMSEG,IBEGIN,CYCIN,XAXIS)
C
C  THIS SUBROUTINE IS CALLED FROM TSERPLOT2 TO
C  PRODUCE PLOTS OF ANY FEB VARIABLE VS CYCLE NUMBER.
C
C  JIM VEGA     CSC     SEPT. 1981
C
C  **********************************************************
C
C  COMMONS FOR ZREAD
      COMMON/RHDR/LR,NP,NBR,NMBR,NMFR,NFR,NIR,NAR,IPR(15)
      COMMON/RDOCF/FDOCR(40)/RDOCI/IDOCR(20)/RDOCA/ADOCR(50)
      COMMON/RDATA/VR(15,500)
      COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
C
      COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IDEC,YORIG
C
      DIMENSION XARAY(1000),YARAY(1000)
      DIMENSION LAB(7),LBLX(3)
C
C
      INDX=1
      NCYCLE=1
      IB=IBEGIN
      IFIRST=1
      IPLT=0
      LSTSEG=NUMSEG+IBEGIN-1
      XDEC=IDEC+1.0
C
  20  CONTINUE
      IF (IFIRST.EQ.1) GO TO 25
      IF (NCYCLE.LT.NR) GO TO 30
  25  CALL ZREAD (IU,IF,IB)
      NCYCLE=1
      IF (IFIRST.EQ.1.AND.IF.NE.0) GO TO 998
      IF (ITERM.EQ.1) GO TO 100
      IF (IF.NE.0) GO TO 100
      IB=IB+1
C
  30  CONTINUE
      IF (IFIRST.NE.1) GO TO 50
C  INITIALIZE SUBPLOT ON FIRST PASS
      ENCODE (13,3000,LBLX)
3000  FORMAT ('CYCLE NUMBERS')
      CALL XINTAX
      CALL LABEL (IPR(IR),NMBR,FDOCR(10),IDOCR(11),IP,LAB,LBLY)
      CALL PHYSOR (1.25,YORIG)
      CALL TITLE (LAB,100,LBLX,100,LBLY,6,XAXIS,4.5)
      XMIN=IBEGIN*500.-499.
      XSTP=CYCIN*2.0
      XMAX=NUMSEG*500.+XMIN
      CALL GRAF (XMIN,XSTP,XMAX,YMIN,YSTP,YMAX)
```

```
      C
      C  FRAME SUBPLOT
            XLNGTH=XAXIS*2.0
            CALL STRTPT (-1.C0,-.75)
            CALL CONNPT (-1.0,5.25)
            CALL CONNPT (XLNGTH,5.25)
            CALL CONNPT (XLNGTH,-.75)
            CALL CONNPT (-1.C0,-.75)
      C
      C
            IFIRST=0
      C
      50    CONTINUE
      C  SET UP AXIS ARRAYS
            XARAY(INDX)=XDEC*INDX-(XDEC-1.)+IPLT*5000.
            YARAY(INDX)=VR(IP,NCYCLE)
      C
            INDX=INDX+1
            NCYCLE=NCYCLE+IDEC
      C
            IF (IDOCR(4).EQ.LSTSEG) ITERM=1
            IF (IDOCR(1).EQ.1) ITERM=1
      C
            IF (INDX.LT.1001) GO TO 20
      C
            INDX=INDX-1
            CALL CURVE (XARAY,YARAY,INDX,0)
            INDX=1
            IPLT=IPLT+1
            IF (ITERM.NE.1) GO TO 20
      C
      100   CONTINUE
      C  TERMINATE SUBPLOT
            CALL CURVE (XARAY,YARAY,INDX,0)
            CALL ENDGR (0)
            GO TO 1000
      998   WRITE (6,999) IF
      999   FORMAT (' ERROR IN ZREAD, IF=',I4)
      1000  CONTINUE
            RETURN
            END

@PRT,S V.RELSEC/PLOT2

6A*LIB(1).RELSEC/PLOT2
            SUBROUTINE RELSEC (ISTSEC,INDSEC,IPTIME)
      C
      C  THIS SUBROUTINE IS CALLED TO PLOT ANY FER
      C  VARIABLE AGAINST RELATIVE TIME (RELSEC).
      C
      C
      C  JIM VEGA     CSC     SEPT. 1981
      C
      C  ***************************************************************
      C
      C  COMMONS FOR ZREAD
            COMMON/RHDR/LR,NR,NBR,NMBR,NMFR,NFR,NIR,MR,IPF(15)
            COMMON/RDOCF/FDOCR(50)/FDOCI/IDOCR(50)/RDOCAC/ACR(100)
            COMMON/RDATA/VR(15,500)
            COMMON/DIAGS/MSGR,MSGW,NNNR,NNNW,NNIP,NNF,NNI,NNA,IRST,IWST
      C
            COMMON/PLOT/IU,IP,YMAX,YSTP,YMIN,IDEC,YORIG
```

```
      C
            DIMENSION XARAY(1000),YARAY(1000)
            DIMENSION LAR(7),LBLX(3)
      C     WRITE (6,1114) IU,IP,YMAX,YSTP,YMIN,IDEC
      C1114  FORMAT (2(I5),2X,3(F7.2,3X),I6)
      C
            IB=0
            ITERM=0
            NCYCLE=0
            ICOUNT=0
            INDX=1
            IFIRST=1
            STRTSC=ISTSEC+1.
        96  WRITE(6,99)
        99  FORMAT(' NUMBER OF SECONDS PER INCH?')
            READ(5,98)TINC
        98  FORMAT()
            XLNGTH=(INDSEC-ISTSEC)/TINC+1.0
            IF(XLNGTH.GT.71.)WRITE(6,97)
        97  FORMAT(' X AXIS TOO LONG (MAX=71):  REENTER')
            IF(XLNGTH.GT.71.)GO TO 96
      C
       20   CONTINUE
            IB=IB+1
            CALL ZREAD (IU,IF,IB)
            IF (IF.NE.0) GO TO 998
            IF (IFIRST.EQ.0) GO TO 25
      C  INITIALIZE SUBPLOT ON FIRST PASS
            ENCODE (7,3000,LBLX) IPS(15)
      3000  FORMAT (A6,'S')
            CALL XINTAX
            CALL LABEL (IPR(IP),NMBR,FDOCR(10),IDOCR(11),IP,LAR,LBLY)
            CALL PHYSOR (1.25,YORIG)
      C     WRITE (6,1113) YORIG
      C1113  FORMAT (F7.3)
            CALL TITLE (LAR,100,LBLX,100,LBLY,6,XLNGTH,4.5)
            CALL YGRAF (STRTSC,TINC,YMIN,YSTP,YMAX)
            IFIRST=0
            NR1=NR-IDEC
      C
       25   CONTINUE
            IF (.NOT.(ISTSEC.GE.VR(IPTIME,1).AND.ISTSEC.LE.VR(IPTIME,NR)))
          $ GO TO 20
      C
       30   CONTINUE
            NCYCLE=NCYCLE+1
      C
            IF (ISTSEC.GE.VR(IPTIME,NCYCLE)) GO TO 30
      C
       60   CONTINUE
            DELTAT=VR(IPTIME,NCYCLE+IDEC)-VR(IPTIME,NCYCLE)
            IF (DELTAT.GE.5.0) GO TO 200
            IF (VR(IPTIME,NCYCLE).GE.INDSEC) GO TO 100
      C
      C  SET UP AXIS ARRAYS
            XARAY(INDX)=VR(IPTIME,NCYCLE)
            YARAY(INDX)=VR(IP,NCYCLE)
      C     WRITE (6,1111)VR(IPTIME,NCYCLE),VR(IP,NCYCLE
      C    $ XARAY(INDX),YARAY(INDX)
      C1111  FORMAT (4(F7.2,3X))
      C
      C     WRITE(6,90)IPTIME,NCYCLE,IP
            IF (.NOT.NCYCLE.GE.NR1) GO TO 60
```

```
              IB=IB+1
              CALL ZREAD (IU,IF,IB)
              IF (IF.NE.0) GO TO 100
              NCYCLE=1-IDEC
      C
      60      CONTINUE
              IF (INDX.EQ.1000) GO TO 200
      C
      70      CONTINUE
              NCYCLE=NCYCLE+IDEC
              INDX=INDX+1
              GO TO 50
      C
      C   **********************************************************************
      C
      C   PLOT DATA
      C
      100     CONTINUE
              ITERM=1
      C
      200     CONTINUE
              IF (INDX.NE.1000) INDX=INDX-1
              WRITE (6,1112) ITERM,INDX
      1112    FORMAT (' CALLING CURVE   ITERM=',I3,'   INDX=',I6)
              CALL CURVE (XARAY,YARAY,INDX,0)
              ICOUNT=ICOUNT+INDX
              INDX=0
      C
              IF (ITERM.NE.1) GO TO 70
              XFRAME=XLNGTH+2.5
      C
      C   FRAME SUBPLOT
              CALL STRTPT (-1.00,-.75)
              CALL CONNPT (-1.0,5.25)
              CALL CONNPT (XFRAME,5.25)
              CALL CONNPT (XFRAME,-.75)
              CALL CONNPT (-1.00,-.75)
      C
              CALL ENDGR (0)
              GO TO 1000
      C
      996     WRITE (6,999) IF
      999     FORMAT (' ERROR IN ZREAD, IF=',I4)
      C
      1000    CONTINUE
              RETURN
              END
```

```
@PRT,S V.LABEL/PLOT2

;A*LIB(1).LABEL/PLOT2
              SUBROUTINE LABEL (NAMVAR,NMBR,DAY,IYR,IP,LAB,L3LY)
      C
      C   THIS SUBROUTINE ENCODES THE DATA LITERALS USED FOR
      C   TITLES ON THE PLOTS PRODUCED BY TSERPLOT2.
      C
      C   JIM VEGA      CSC      SEPT. 1981
      C
      C   ***************************************************************
      C
      C   SUBPLOT TITLE
            ENCODE (40,3000,LAB) NAMVAR,NMBR,DAY,IYR
```

```
      C
      C  ORDINATE TITLE
            IF (IP.EQ.3.OR.IP.EQ.14) ENCODE(6,3010,LBLY) NAMVAR
            IF (IP.EQ.1.OR.IP.EQ.2.OR.IP.EQ.4.OR.IP.EQ.8.OR.
           $ IP.EQ.9.OR.IP.EQ.10) ENCODE(6,3011,LBLY) NAMVAR
            IF (IP.EQ.5.OR.IP.EQ.6.OR.IP.EQ.7.OR.IP.EQ.11.OR.
           $ IP.EQ.12.OR.IP.EQ.13.OR.IP.EQ.15) ENCODE(7,3012,LBLY) NAMVAR
      C
            RETURN
      3000  FORMAT (A6,'   CAST  ',A3,'  ',F8.3,'  'I4,'S')
      3010  FORMAT (A4,'  ')
      3011  FORMAT (A5,' ')
      3012  FORMAT (A6)
            END

@PRT,S V.MAPPLOT2
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: NORDA Technical Note 155

2. GOVT ACCESSION NO.: AD-A121 498

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): Documentation of Software for the Neil Brown Instrument Systems/NORDA Velocity/CTD Profiler

5. TYPE OF REPORT & PERIOD COVERED: Final

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s):
K.D. Saunders
Henry Perkins

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS:
Naval Ocean Research and Development Activity
NSTL Station, Mississippi 39529

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS:
Naval Ocean Research and Development Activity
NSTL Station, Mississippi 39529

12. REPORT DATE: July 1982

13. NUMBER OF PAGES: 61

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report):
Distribution Unlimited

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
oceanographic data processing
UNIVAC 1108
profiler
velocity
CTD

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
Software developed for processing Neil Brown Instrument Systems/NORDA profiler data is documented in this report. The software includes programs for translating the profiler data from original NBIS format to engineering units in UNIVAC/NAVOCEANO FEB files and for editing and correcting the data subsequently. This report provides complete descriptions of the programs as well as operating information.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)